

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ  
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**Дисциплина:** Фронт-энд разработка

Отчет

Лабораторная работа 3: Разработка одностраничного  
веб-приложения (SPA) с использованием фреймворка  
Vue.JS

Выполнил:  
Чан Дык Минь

Группа: K33392

Проверил:  
Добряков Д. И.

Санкт-Петербург

2023 г.

## Задача

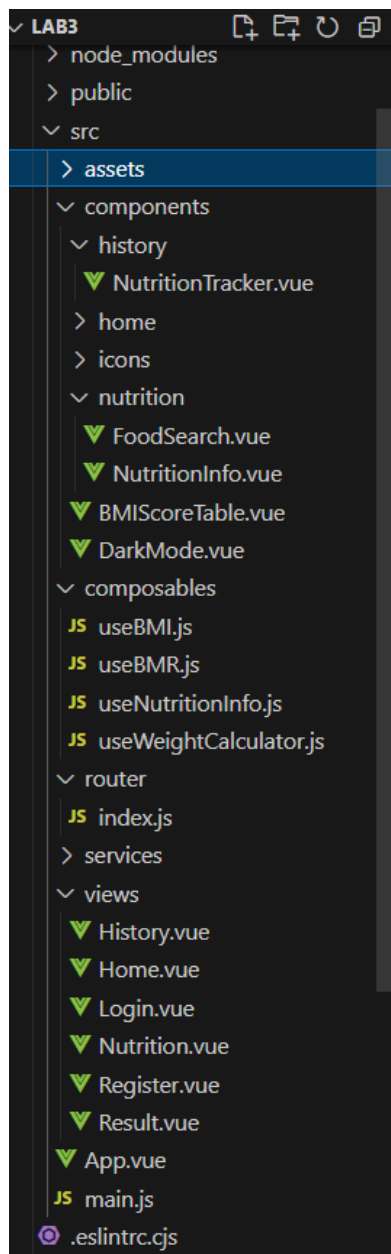
Мигрировать ранее написанный сайт на фреймворк Vue.JS.

Минимальные требования:

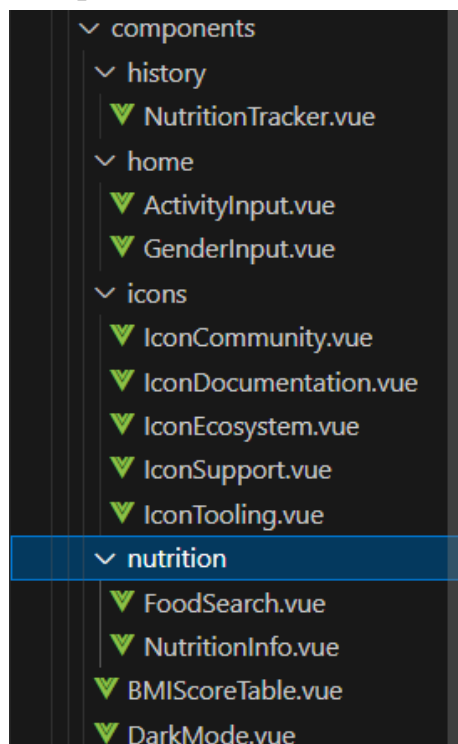
- Должен быть подключён роутер
- Должна быть реализована работа с внешним API
- Разумное деление на компоненты
- Использование composable

## Ход работы

### 1. Структура проекта:



Components:



Чтобы проект был понятен, а код выглядел чище и управляемее, я разделил страницы на небольшие компоненты. Это облегчит задачу редактирования страницы.

## 2. Связь с внешним API:

Наш сайт использует API FoodData Central Министерства сельского хозяйства США для получения информации о пищевой ценности продуктов. Центральный API USDA FoodData предоставляет подробные данные о пищевой ценности тысяч продуктов питания, включая калории, белки, углеводы, жиры и многие другие минералы.

Реализация вызова API:

Мы используем библиотеку Axios в Vue.js для вызовов API. Вот пример того, как мы делаем вызов API в Composable:

```

1  import { ref } from 'vue';
2  import axios from 'axios';
3
4  const apiKey = 'e5FpDu9gY6PdfuHULvHyrHKvPfnKr0U4DkaFlYrX';
5
6  export function useNutritionInfo() {
7    const food = ref("");
8    const nutritionInfo = ref({});
9
10   async function getNutritionInfo(foodName) {
11     const apiUrl = `https://api.nal.usda.gov/fdc/v1/foods/search?query=${foodName}&api_key=${apiKey}`;
12     try {
13       const response = await axios.get(apiUrl);
14       const data = response.data;
15
16       if (data.foods && data.foods.length > 0) {
17         const food = data.foods[0];
18         const nutrients = food.foodNutrients;
19         nutritionInfo.value = {
20           calories: nutrients.find(n => n.nutrientName === "Energy")?.value || 0,
21           protein: nutrients.find(n => n.nutrientName === "Protein")?.value || 0,
22           fat: nutrients.find(n => n.nutrientName === "Total lipid (fat)")?.value || 0,
23           carbs: nutrients.find(n => n.nutrientName === "Carbohydrate, by difference")?.value || 0,
24         };
25       } else {
26         console.log("No nutrition information found for", foodName);
27       }
28     } catch (error) {
29       console.error('Error sending API request: ', error);
30     }
31   }
32
33   return { food, nutritionInfo, getNutritionInfo };
34 }

```

Показать результаты:

После того, как мы получим данные от API, мы используем их в компоненте для отображения информации о пищевой ценности

```

1  <template>
2    <div class="container mt-3" id="nutritionInfo">
3      <h3 class="text-#282828 dark:text-#bbbbbb">Nutrition Information for {{ foodName }} (per 100g)</h3>
4      <p>Calories: {{ nutritionInfo.calories }} kcal</p>
5      <p>Protein: {{ nutritionInfo.protein }} g</p>
6      <p>Fat: {{ nutritionInfo.fat }} g</p>
7      <p>Carbs: {{ nutritionInfo.carbs }} g</p>
8    </div>
9  </template>
10
11  <script setup>
12    const { props } = defineProps(['foodName', 'nutritionInfo']);
13  </script>
14
15  <style>
16    h2 {
17      color: hsla(160, 100%, 37%, 1);
18    }
19  </style>

```

### 3. Использование Composables

Использование Composables во Vue 3 дает множество преимуществ:

**Связанная логическая организация:**

Составные элементы помогают организовать связанную логику в естественных объектах, таких как ИМТ, BMR, NutritionInfo,

WeightCalculator. Каждый Composable ориентирован на конкретную задачу, что делает исходный код простым для чтения и понимания.

### Управление статусом:

Composables помогают эффективно управлять состоянием с помощью реактивных и ref-функций Vue 3.

Такие статусы, как информация о питании, вес, ИМТ, управляются гибко.

### Повторное использование исходного кода:

Составные элементы можно использовать повторно, что помогает уменьшить дублирование кода и способствует реализации принципа DRY (не повторяйте себя).

Если у вас есть несколько компонентов, которым необходимо использовать одну и ту же логику (например, расчет ИМТ), вы можете просто импортировать и использовать соответствующий Composable.

#Nutrition.vue

```
<div class="flex flex-col">
  <FoodSearch :getFoodName="getFoodName" />
  <NutritionInfo :foodName="food" :nutritionInfo="nutritionInfo" />
</div>
```

```
<script setup>
import { useNutritionInfo } from '@/composables/useNutritionInfo';
import FoodSearch from '../components/nutrition/FoodSearch.vue';
import NutritionInfo from '../components/nutrition/NutritionInfo.vue';

const { food, nutritionInfo, getNutritionInfo } = useNutritionInfo();

const getFoodName = async (foodName) => {
  await getNutritionInfo(foodName);
};
</script>
```

## #FoodSearch.vue

```
<template>
  <div class="container mt-2 flex flex-row items-center ">
    <!-- Thêm một lớp mới hoặc điều chỉnh lớp hiện tại -->
    <input v-model="food" type="text" class="form-control smaller-input input-food " placeholder="Search your food">
    <button class="btn mx-2 px-3" :class="{ 'btn-outline-light': isDark, 'btn-outline-dark': !isDark }" type="button" @click="handleClick">
      Show info
    </button>
  </div>
</template>

<script setup>
import { ref, defineProps } from 'vue';
import { isDark, toggleDark } from '@services/DarkModeService';

const props = defineProps(['getFoodName']);
const food = ref("");

const handleClick = () => {
  props.getFoodName(food.value);
};
</script>
```

## #NutritionInfo.vue

```
<template>
  <div class="container mt-3" id="nutritionInfo">
    <h3 class="□text-[#282828] ■dark:text-[#bbbbbb]">Nutrition Information for {{ foodName }} (per 100g)</h3>
    <p>Calories: {{ nutritionInfo.calories }} kcal</p>
    <p>Protein: {{ nutritionInfo.protein }} g</p>
    <p>Fat: {{ nutritionInfo.fat }} g</p>
    <p>Carbs: {{ nutritionInfo.carbs }} g</p>
  </div>
</template>

<script setup>
const { props } = defineProps(['foodName', 'nutritionInfo']);
</script>
```

## Составные части:

useNutritionInfo: это Composable, используемый для управления состоянием, связанным с информацией о питании. Я использовал его, чтобы получить информацию о пищевой ценности из API Central FoodData Министерства сельского хозяйства США.

В useNutritionInfo я использовал ссылки для создания ссылочных переменных, которые могут отслеживать изменения, и использовал API композиции для обработки логики и подключения к API.

## Компоненты:

Питание: это основной компонент, содержащий заголовок и описание приложения. Я интегрировал сюда FoodSearch и NutritionInfo.

FoodSearch: небольшой компонент для поиска еды. Я использовал defineProps, чтобы получить свойство getFoodName из родительского компонента, и использовал ссылку для управления состоянием поля ввода.

NutritionInfo: этот компонент отображает информацию о пищевой ценности продукта. Я получил имя еды и информацию о питании от родительского компонента через опору.

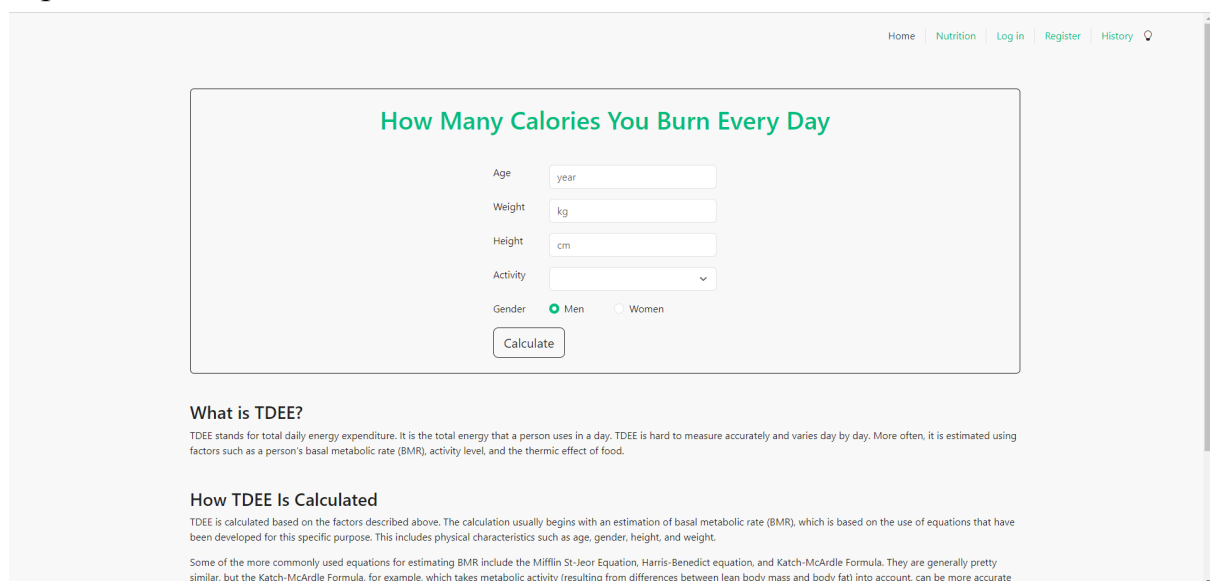
### Соединение между компонентами:

Я использовал реквизит для передачи данных между родительским компонентом и дочерними компонентами. Например, FoodSearch получает свойство getFoodName от родительского компонента Nutrition.

### Обработка событий и вызовы API:

В FoodSearch я использовал v-модель, чтобы связать поле ввода с переменной food. Событие @click запускается при нажатии кнопки и вызывает handleClick.

В разделе «Питание» я использовал Composable useNutritionInfo, чтобы получить информацию о пищевой ценности на основе названий продуктов, переданных из FoodSearch.



Home | Nutrition | Log in | Register | History

### How Many Calories You Burn Every Day

Age

Weight

Height

Activity

Gender ☒ Men ☐ Women

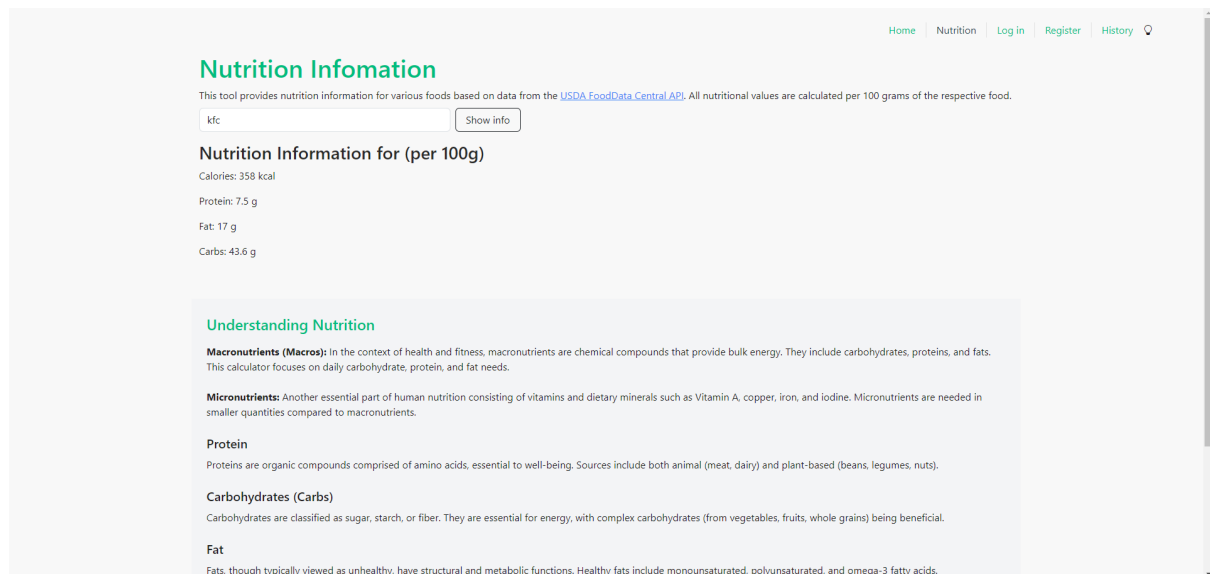
#### What is TDEE?

TDEE stands for total daily energy expenditure. It is the total energy that a person uses in a day. TDEE is hard to measure accurately and varies day by day. More often, it is estimated using factors such as a person's basal metabolic rate (BMR), activity level, and the thermic effect of food.

#### How TDEE Is Calculated

TDEE is calculated based on the factors described above. The calculation usually begins with an estimation of basal metabolic rate (BMR), which is based on the use of equations that have been developed for this specific purpose. This includes physical characteristics such as age, gender, height, and weight.

Some of the more commonly used equations for estimating BMR include the Mifflin St-Jeor Equation, Harris-Benedict equation, and Katch-McArdle Formula. They are generally pretty similar, but the Katch-McArdle Formula, for example, which takes metabolic activity (resulting from differences between lean body mass and body fat) into account, can be more accurate.



## Вывод

Я узнал, как развернуть веб-проект с помощью платформы VueJS, используя маршрутизатор для переключения между страницами. Кроме того, я также научился разделять на небольшие компоненты, писать составные элементы для более удобного использования и управления проектами.