

Разработка клиент-серверного веб-приложения для учёта учебного времени: Vue.js + Node.js

Саяпин Никита Анатольевич

Приамурский государственный университет им.Шолом-Алейхема

Студент

Аннотация

В работе рассматривается процесс разработки и реализации клиент-серверного веб-приложения для учёта учебного времени с использованием технологий Vue.js и Node.js. Приводится структура проекта, включающая серверную часть с REST-интерфейсом, реализованную на платформе Node.js с применением фреймворка Express и базы данных lowdb, а также клиентскую часть на Vue.js, обеспечивающую динамическое обновление данных и визуализацию статистики с помощью библиотеки Chart.js.

Ключевые слова: Vue.js, Node.js, REST API, lowdb, веб-приложение, клиент-сервер, визуализация, учебный трекер.

Development of a client-server web application for accounting of study time: Vue.js + Node.js

Sayapin Nikita Anatolyevich

Sholom-Aleichem Priamursky State University

Student

Abstract

This article presents the development and implementation of a client-server web application for tracking study time using Vue.js and Node.js technologies. The project architecture includes a server part with a REST interface built on Node.js and Express framework with lowdb database, and a client part based on Vue.js providing dynamic data updates and visualization with Chart.js. It is shown that combining Vue.js and Node.js allows creating a lightweight, extensible, and visually clear application suitable for educational and research purposes. The project demonstrates the practical use of client-server architecture, data processing via API, and modern web development tools.

Keywords: Vue.js, Node.js, REST API, lowdb, web application, client-server, visualization, training tracker.

1. Введение

1.1 Актуальность

Современные образовательные инструменты всё чаще требуют не просто статических страниц, а динамических интерфейсов с реактивными обновлениями и хранением данных. Использование клиент-серверной

архитектуры с REST API даёт гибкость: фронтенд остаётся лёгким и отзывчивым, а сервер управляет логикой и хранением данных. Vue.js как прогрессивный JavaScript-фреймворк широко используется для разработки SPA (single-page applications) благодаря простоте, лёгкости обучения и богатому экосистемному окружению. В сочетании с Node.js и Express такой стек часто применяют в веб-проектах среднего масштаба.

Для учебных проектов важно создать пример, который: показывает весь цикл от клиента до сервера, не требует сложной СУБД (использует JSON-базу), содержит визуализацию результатов (диаграммы).

Таким образом, статья будет полезна студентам и начинающим разработчикам как инструкция и демонстрация архитектурных решений.

1.2 Обзор исследований

В работе Д.А. Омельченко подробно описывается реализация одностраничного приложения на связке Django + Vue.js, обсуждаются особенности SPA-архитектуры, работа канала обмена и клиентской части на Vue, что напрямую соотносится с нашей фронтенд-частью и подходом к реактивному интерфейсу и маршрутизации данных между браузером и сервером. Показанные приёмы интеграции Vue с сервером подтверждают устойчивость выбранной нами схемы взаимодействия и облегчают перенос идей на любой REST-бэкенд, включая Node/Express [1]. В.Н. Волков рассматривает преимущества и ограничения фреймворка Vue.js для построения пользовательских интерфейсов, акцентируя внимание на компонентной модели, двустороннем связывании и низком пороге входа. Эти выводы объясняют, почему Vue целесообразно применять в учебных проектах с быстрыми итерациями и визуализацией (у нас — Chart.js), а также поддерживают сделанный выбор технологического стека для клиентской части [2]. В.К. Баёв демонстрирует практическую разработку веб-сервиса распознавания цифр с клиентом как SPA на Vue.js и серверной частью, обрабатывающей запросы с клиента. Работа иллюстрирует полный цикл «запрос–обработка–ответ» и показывает, как Vue-клиент интегрируется с сервером для вычислительных задач — подход аналогичен нашему обмену через REST API, где фронтенд запрашивает списки и агрегаты, а бэкенд отвечает JSON-данными [3]. А.В. Артёмов на примере менеджера закладок показывает связку React на фронтенде и Express на сервере; хотя клиент реализован на другом фреймворке, сама логика REST-маршрутов, организация CRUD и роль Express как универсального слоя маршрутизации полностью соответствуют нашему серверному модулю. Материал служит внешним подтверждением типовой архитектуры CRUD-веб-приложения на Node/Express, применяемой и в нашей лабораторной работе [4]. Д.С.Гржибовский систематизирует подходы к разработке серверных приложений на Node.js и подчёркивает место Express как «стандартного» инструмента для веб-слоя. Представленная классификация и практические замечания по построению серверной логики, структуре маршрутов и обработке запросов подкрепляют выбранную нами реализацию REST-

интерфейса и подтверждают корректность принятых архитектурных решений [5].

1.3 Цель исследования

Целью исследования является демонстрация разработки и функционирования клиент-серверного веб-приложения, реализующего учёт учебного времени на основе современного технологического стека Vue.js и Node.js. В рамках работы решается задача создания архитектуры, включающей серверную часть с REST-интерфейсом, реализованным на платформе Node.js с использованием фреймворка Express и базы данных lowdb, а также клиентскую часть на Vue.js, обеспечивающую визуализацию данных, фильтрацию, добавление и удаление записей без перезагрузки страницы.

2 Материалы и методы

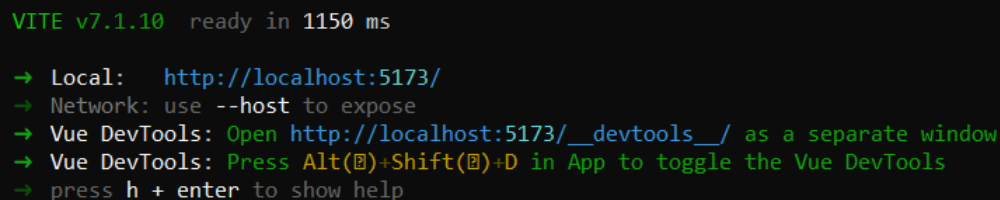
В качестве основной среды разработки используется язык программирования JavaScript, применяемый как для клиентской, так и для серверной части приложения. Серверная часть реализуется на платформе Node.js с использованием фреймворка Express, который обеспечивает обработку HTTP-запросов, маршрутизацию и реализацию REST-интерфейса. В качестве базы данных используется lowdb — лёгкое JSON-хранилище, не требующее установки отдельной СУБД и позволяющее хранить данные в локальном файле.

2.1 Подготовка

Для создания клиентской части приложения используется современный инструмент сборки Vite, обеспечивающий быструю и удобную инициализацию проекта на Vue.js версии 3. На рабочую станцию устанавливается среда Node.js актуальной LTS-версии, включающая менеджер пакетов npm, который применяется для установки зависимостей и запуска проекта.

В терминале выполняется команда `npm create vue@latest`, инициирующая создание нового проекта Vue. В процессе инициализации выбираются базовые настройки по умолчанию, что позволяет получить минимальную структуру, включающую исходные файлы и конфигурацию сборки. После завершения генерации проекта выполняется переход в созданный каталог и установка зависимостей с помощью команды `npm install`.

Запуск приложения осуществляется командой `npm run dev`, после чего локальный сервер Vite автоматически открывает проект в браузере по адресу `http://localhost:5173`. При успешной инициализации на экране отображается стартовая страница Vue, подтверждающая корректность установки и готовность среды к дальнейшей разработке (Рис. 1).



```
VITE v7.1.10 ready in 1150 ms

→ Local:   http://localhost:5173/
→ Network: use --host to expose
→ Vue DevTools: Open http://localhost:5173/__devtools__/ as a separate window
→ Vue DevTools: Press Alt(⌘)+Shift(⌘)+D in App to toggle the Vue DevTools
→ press h + enter to show help
```

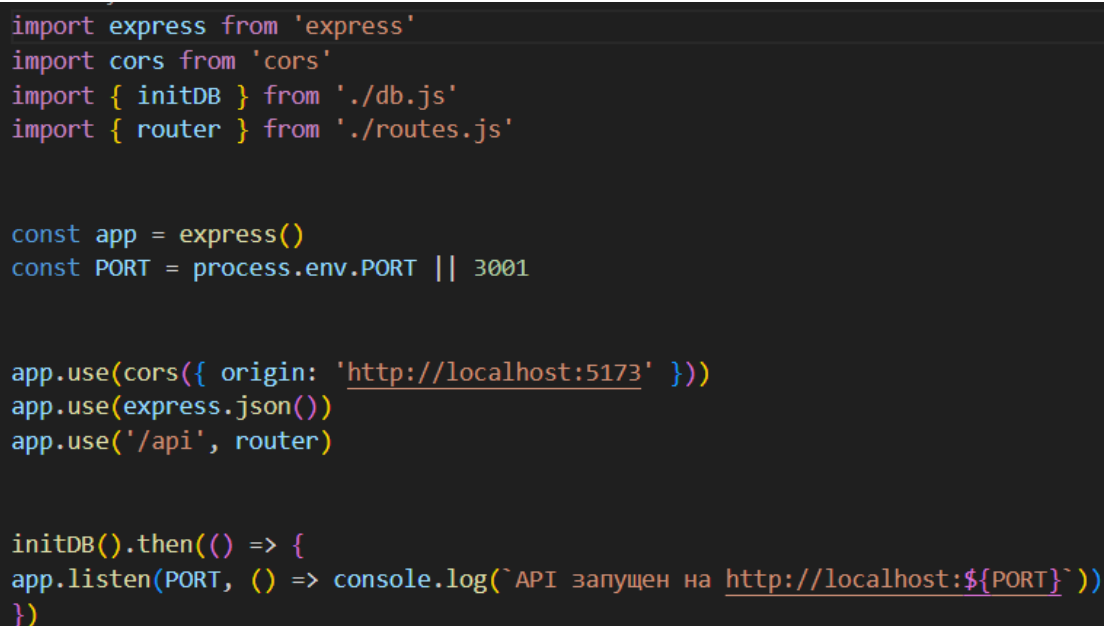
Рисунок 1 – окно создания проекта на vue

Для написания кода используется интегрированная среда Visual Studio Code, обеспечивающая подсветку синтаксиса, автодополнение и встроенный терминал для запуска команд. В каталоге src создаются основные файлы приложения: App.vue, main.js и папка components для пользовательских компонентов. Такая структура упрощает расширение проекта и дальнейшую интеграцию клиентской части с сервером.

3. Результаты

На данном этапе выполняется реализация и запуск проекта, включающего серверную и клиентскую части. После установки зависимостей и подготовки структуры выполняется настройка серверной части приложения. Сервер создаётся на платформе Node.js с использованием фреймворка Express, обеспечивающего маршрутизацию и обработку HTTP-запросов. Для хранения данных применяется библиотека lowdb, позволяющая работать с локальным файлом формата JSON без необходимости установки внешней СУБД.

Файл index.js содержит основной код запуска сервера, включая подключение маршрутов, активацию CORS и обработку данных в формате JSON (Рис. 2).



```
import express from 'express'
import cors from 'cors'
import { initDB } from './db.js'
import { router } from './routes.js'

const app = express()
const PORT = process.env.PORT || 3001

app.use(cors({ origin: 'http://localhost:5173' }))
app.use(express.json())
app.use('/api', router)

initDB().then(() => {
  app.listen(PORT, () => console.log(`API запущен на http://localhost:${PORT}`))
})
```

Рисунок 2 – Код index.js

Отдельно создаются файлы `db.js` и `routes.js`, реализующие логику взаимодействия с базой данных и обработку запросов API (Рис. 3).

```
import { Low } from 'lowdb'
import { JSONFile } from 'lowdb/node'
import { join, dirname } from 'node:path'
import { fileURLToPath } from 'node:url'

const __dirname = dirname(fileURLToPath(import.meta.url))
const file = join(__dirname, 'db.json')
const adapter = new JSONFile(file)
export const db = new Low(adapter, { sessions: [] })

export async function initDB() {
  await db.read()
  db.data ||= { sessions: [] }
  await db.write()
}
```

Рисунок 3 – Создание локальной базы данных

После запуска сервера командами `npm install` и `npm run dev` серверное приложение начинает работать на порту 3001, ожидая входящих запросов от клиента (Рис. 4).

```
> vue-tracker-server@1.0.0 dev
> node index.js

API запущен на http://localhost:3001
```

Рисунок 4 – Запуск серверной части проекта и работа API на порту 3001

В браузере или инструменте Postman можно проверить корректность функционирования API-запросов. При обращении к маршруту `http://localhost:3001/api/sessions` возвращается JSON-массив текущих записей учебных занятий. Создание, обновление и удаление записей выполняется через соответствующие маршруты POST, PUT и DELETE, что демонстрирует работу REST-интерфейса и корректную обработку запросов.

Клиентская часть реализуется на фреймворке Vue.js, который обеспечивает реактивное обновление интерфейса и мгновенный отклик при изменении данных. В файле `App.vue` задаётся структура основного

компонента приложения, в котором отображаются форма добавления, таблица с записями и диаграмма прогресса. Для взаимодействия с сервером создаётся модуль `api.js`, содержащий функции для отправки запросов и получения ответов от REST API.

Компонент `SessionForm.vue` отвечает за ввод новых данных. Он содержит поля для выбора даты, ввода названия предмета, количества минут и комментария. После нажатия кнопки «Добавить» выполняется вызов функции `createSession()`, которая отправляет данные на сервер. При успешном выполнении запроса сервер возвращает обновлённый список занятий, а интерфейс мгновенно обновляется (Рис. 5).

```
<template>
  <div class="card">
    <h2>Добавить занятие</h2>
    <form @submit.prevent="submit" class="grid" style="grid-template-columns: repeat(4, 1fr); gap: 12px;">
      <div>
        <label>Дата</label>
        <input type="date" v-model="form.date" required>
      </div>
      <div>
        <label>Предмет</label>
        <input type="text" v-model.trim="form.subject" placeholder="Математика" required>
      </div>
      <div>
        <label>Минуты</label>
        <input type="number" min="1" step="1" v-model.number="form.minutes" required>
      </div>
      <div>
        <label>Заметка</label>
        <input type="text" v-model.trim="form.note" placeholder="Домашнее задание">
      </div>
      <div style="grid-column: 1 / -1; display:flex; gap:10px; align-items:center;">
        <button class="btn">Добавить</button>
        <span class="badge">Данные хранятся на сервере</span>
      </div>
    </form>
  </div>
</template>
```

Рисунок 5 – Форма добавления записи в приложении

Отображение сохранённых записей реализуется в компоненте `SessionTable.vue`. Таблица содержит фильтры по дате и предмету, а также возможность сортировки по длительности занятий. При удалении записи срабатывает функция `deleteSession()`, которая вызывает соответствующий метод API и обновляет таблицу без перезагрузки страницы. Реактивная модель данных Vue обеспечивает немедленное обновление интерфейса после каждого изменения (Рис. 6).

```

<template>
  <div class="card">
    <div style="display:flex; justify-content:space-between; align-items:center;">
      <h2>Занятия ({{ items.length }})</h2>
      <div>Всего минут: <b>{{ total }}</b></div>
    </div>
    <div style="display:grid; grid-template-columns: repeat(4, 1fr); gap: 12px; margin-top:8px;">
      <input placeholder="Фильтр по предмету" v-model="q">
      <input type="date" v-model="from">
      <input type="date" v-model="to">
      <select v-model="sort">
        <option value="date_desc">По дате (новые → старые)</option>
        <option value="date_asc">По дате (старые → новые)</option>
        <option value="minutes_desc">По минутам (убыв.)</option>
        <option value="minutes_asc">По минутам (возр.)</option>
      </select>
    </div>

    <div style="overflow:auto; margin-top:12px;">
      <table class="table">
        <thead>
          <tr>
            <th>Дата</th>
            <th>Предмет</th>
            <th>Минуты</th>
            <th>Заметка</th>
            <th></th>
          </tr>
        </thead>
        <tbody>
          <tr v-for="s in filtered" :key="s.id">
            <td>{{ s.date }}</td>
            <td>{{ s.subject }}</td>
            <td>{{ s.minutes }}</td>
            <td>{{ s.note || '-' }}</td>
            <td style="text-align:right">
              <button class="btn danger" @click="$emit('remove', s.id)">Удалить</button>
            </td>
          </tr>
          <tr v-if="filtered.length===0">
            <td colspan="5" style="text-align:center; color:#6b7280">Нет записей</td>
          </tr>
        </tbody>
      </table>
    </div>
  </div>

```

Рисунок 6 – Отображение списка занятий и фильтрация данных

Для визуализации статистики используется компонент `ProgressChart.vue`, основанный на библиотеке `Chart.js`. Диаграмма строится по данным, полученным с маршрута `/api/stats/subjects`, где агрегируется общее количество минут по каждому предмету. График отображает распределение учебного времени за последние семь дней и обновляется автоматически при изменении данных на сервере (Рис. 7).

Суммарное время по предметам (за последние 7 дней)

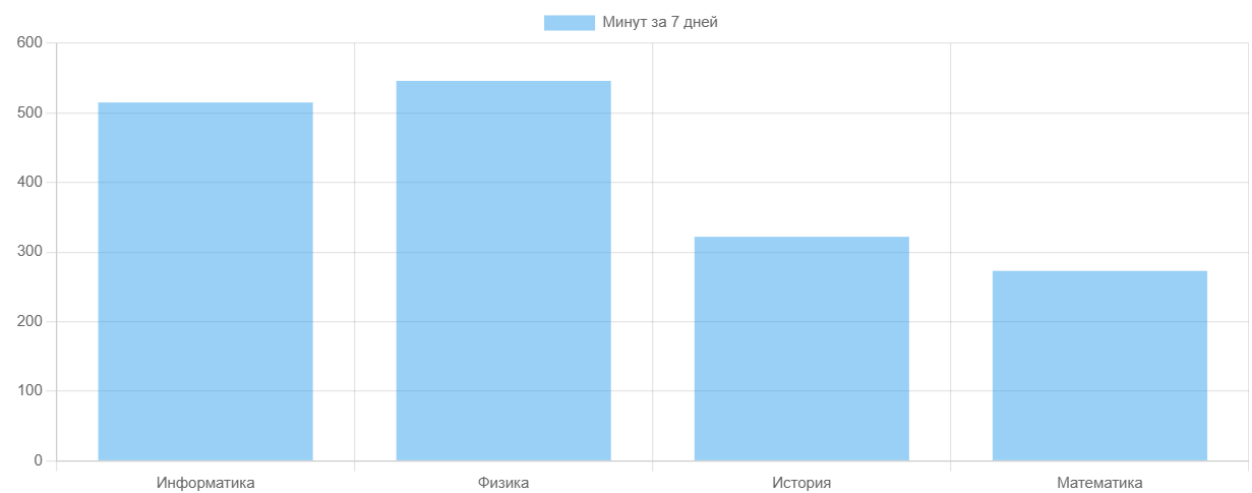


Рисунок 7 – Диаграмма распределения учебного времени по предметам

В результате взаимодействия клиентской и серверной частей реализуется полный цикл работы приложения: пользователь вводит данные, которые сохраняются в базе, обновляются и визуализируются в интерфейсе (Рис.8).

Сохранить

Сгенерировать демо-данные

Добавить занятие

Дата

23.10.2025

Предмет

Математика

Минуты

60

Заметка

Домашнее задание

Добавить

Данные хранятся на сервере

Занятия (24)

Всего минут: 1656

Фильтр по предмету

дд.мм.гггг

дд.мм.гггг

По дате (новые → старые)

Дата	Предмет	Минуты	Заметка	
2025-10-17	Физика	69	демо	Удалить
2025-10-17	Математика	100	демо	Удалить
2025-10-17	Информатика	51	демо	Удалить
2025-10-17	Физика	31	демо	Удалить
2025-10-17	Физика	119	демо	Удалить

Рисунок 8 – Отображение проекта на странице браузера

4 Выводы

Проведённая работа наглядно демонстрирует необходимость системной оценки и обработки данных перед их дальнейшим использованием в аналитических задачах. В ходе эксперимента были выявлены и устранены типичные проблемы, характерные для реальных массивов данных: дубликаты, пропуски, некорректные и выбросные значения. Итоговый набор данных приобрёл целостный и корректный вид, что делает его пригодным для дальнейшего анализа и построения моделей.

Библиографический список

1. Омельченко Д. А., Гонатаев Р. Г., Фешина Е. В. Реализация SPA при помощи Django VUE.js // Гуманитарные и естественно-научные исследования: основные дискуссии. Материалы XXVIII Всероссийской научно-практической конференции. Южный университет (ИУБиП). Издательство ВВМ, 2021. С. 18–22.
2. Волков В. Н., Новиков С. В., Герасимова Н. М., Полонская Т. С. Перспективы использования фреймворка Vue.js для создания гибких пользовательских интерфейсов // Современное программирование. Материалы IV Международной научно-практической конференции. Под общ. ред. Т. Б. Казиахмедова. Нижневартровский государственный университет, 2022. С. 69–74.
3. Баев В. К. Разработка веб-сервиса распознавания цифр на основе многослойных нейронных сетей // Цифровой регион: опыт, компетенции, проекты. ФГБОУ ВО «Брянский государственный инженерно-технологический университет». 2021. С. 63-67.
4. Артемов А. В., Гришунов С. С., Черепков Е. А., Белов Ю. С. Разработка менеджера закладок с использованием библиотек React и Express // Международный студенческий научный вестник. 2019. № 1. С. 42.
5. Гржибовский Д. С. Разработка серверных приложений при помощи Node.js и MongoDB // Научный альманах. 2019. № 11-2(61). С. 28–31.