

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ  
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Фронтенд-разработка

Отчёт

Домашняя работа №5: изучение основ работы с  
менеджером зависимостей npm

Выполнил:

Колмогорова А.С.

К3342

Проверил:

Добряков Д. И.

Санкт-Петербург

2025 г.

## **СОДЕРЖАНИЕ**

1 Задача.....	3
2 Ход работы.....	4
2.1 Инициализация проекта.....	4
2.2 Преднастройка.....	6
2.3 Компоненты, роутинг.....	9
2.4 Работа с v-model, props, v-if, v-for, methods, mounted.....	12
3 Вывод.....	15

## **1 Задача**

В рамках пятого домашнего задания необходимо изучить основные команды пакетного менеджера NPM и научиться стартовать проект на Vue.

Мануал:

<https://docs.google.com/document/d/187UkgGNrcWqkb2aCGpkHTLgeozoEIMqdVgVGMB0C9gk/edit?tab=t.0>

## 2 Ход работы

### 2.1 Инициализация проекта

Первым делом устанавливаем node, npm и vue (последнее – командой со скриншота ниже):

```
C:\Users\sashk>npm install -g @vue/cli
npm warn EBADENGINE Unsupported engine {
npm warn EBADENGINE   package: '@achrinza/node-ipc@9.2.9'
npm warn EBADENGINE   required: {
npm warn EBADENGINE     node: '8 || 9 || 10 || 11 || 12 |'
npm warn EBADENGINE   },
npm warn EBADENGINE   current: { node: 'v24.11.1', npm: ''
npm warn EBADENGINE }
npm warn deprecated source-map-url@0.4.1: See https://git
npm warn deprecated urix@0.1.0: Please see https://github
```

Теперь переходим к инициализации проекта, задаем ему параметры, опции, согласно предложенным.

```
C:\Users\sashk>npm init vue@latest
Need to install the following packages:
create-vue@3.18.3
Ok to proceed? (y) y

> npx
> create-vue

T  Vue.js - The Progressive JavaScript Framework
| 
o  Project name (target directory):
|  vue-project-hw5
|
o  Select features to include in your project: (↑/↓ to navigate, space to select, a to toggle all, enter to confirm)
|  Router (SPA development), Pinia (state management), ESLint (error prevention)
|
o  Select experimental features to include in your project: (↑/↓ to navigate, space to select, a to toggle all, enter to
confirm)
|  none
|
o  Skip all example code and start with a blank Vue project?
|  Yes

Scaffolding project in C:\Users\sashk\vue-project-hw5...
|
- Done. Now run:

  cd vue-project-hw5
  npm install
  npm run dev

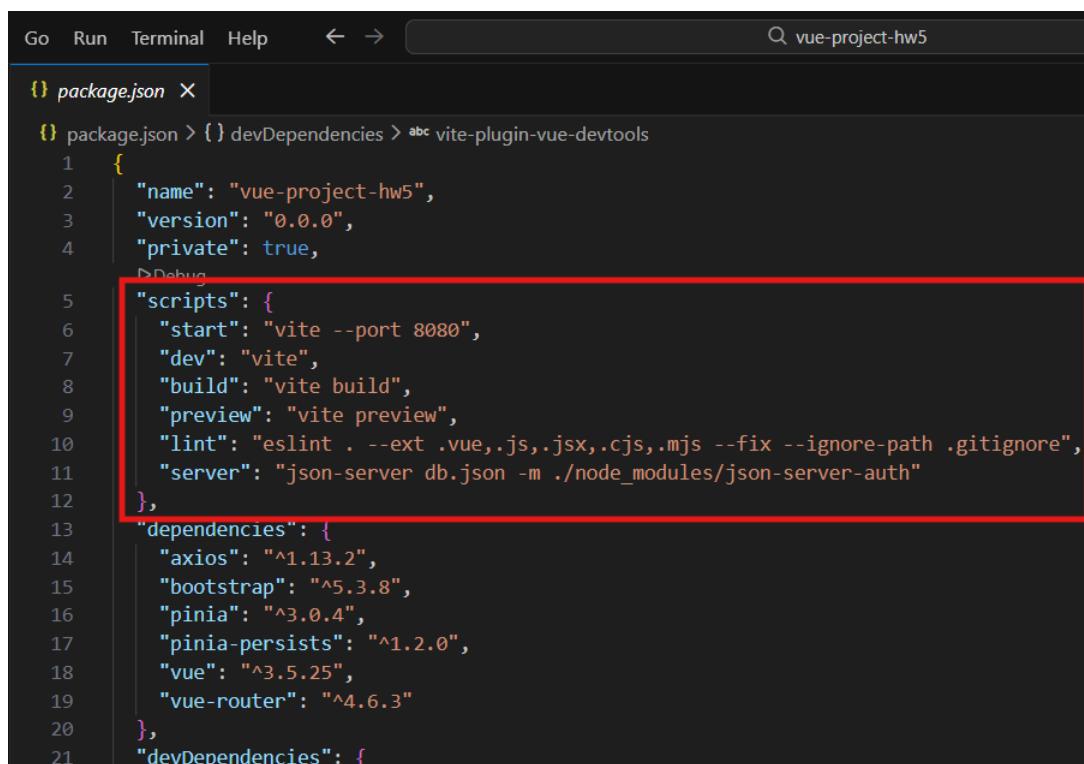
| Optional: Initialize Git in your project directory with:
  git init && git add -A && git commit -m "initial commit"
```

Далее устанавливаем зависимости npm, перейдя в директорию проекта, ожидаем полного создания проекта и переходим в среду разработки для создания и редактирования файлов.

```
C:\Users\sashk>cd vue-project-hw5
C:\Users\sashk\vue-project-hw5>npm install
up to date, audited 229 packages in 1s
57 packages are looking for funding
  run 'npm fund' for details
found 0 vulnerabilities
```

```
C:\Users\sashk\vue-project-hw5>code .
```

Изменяем package.json, добавляя скрипты для старта проекта, подключения компонент (также для подключения сервера и бд). Сохраняем.



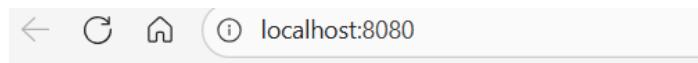
```
{ package.json X
  package.json > {} devDependencies > vite-plugin-vue-devtools
  1  {
  2    "name": "vue-project-hw5",
  3    "version": "0.0.0",
  4    "private": true,
  5    "scripts": {
  6      "start": "vite --port 8080",
  7      "dev": "vite",
  8      "build": "vite build",
  9      "preview": "vite preview",
 10      "lint": "eslint . --ext .vue,.js,.jsx,.cjs,.mjs --fix --ignore-path .gitignore",
 11      "server": "json-server db.json -m ./node_modules/json-server-auth"
 12    },
 13    "dependencies": {
 14      "axios": "^1.13.2",
 15      "bootstrap": "5.3.8",
 16      "pinia": "^3.0.4",
 17      "pinia-persist": "^1.2.0",
 18      "vue": "^3.5.25",
 19      "vue-router": "^4.6.3"
 20    },
 21    "devDependencies": {
```

Проверяем, всё ли работает, выполняя в терминале npm start.

```
C:\Users\sashk\vue-project-hw5>npm start
> vue-project-hw5@0.0.0 start
> vite --port 8080

VITE v7.3.0  ready in 1317 ms

→ Local:  http://localhost:8080/
→ Network: use --host to expose
→ Vue DevTools: Open http://localhost:8080/__devtools__/ as a separate window
→ Vue DevTools: Press Alt(Alt)+Shift(Shift)+D in App to toggle the Vue DevTools
→ press h + enter to show help
```



## You did it!

Visit [vuejs.org](https://vuejs.org) to read the documentation

Всё успешно, поэтому переходим к преднастройке.

## 2.2 Преднастройка

Устанавливаем дополнительные пакеты:

- axios для взаимодействия с внешним API;
- pinia-persist для сохранения стейта проекта в localStorage;
- bootstrap для базовых стилей.

```
C:\Users\sashk\vue-project-hw5>npm i axios pinia-persist bootstrap -S
added 26 packages, and audited 255 packages in 4s

65 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Затем последовательно заполняем файлы в следующих директориях:

- src/api:
  - instance.js – инстанс axios для отлавливания ошибок и инициализации url для сервера

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows a tree view of the project structure under "VUE-PROJECT-HW5". The "src/api" folder contains "instance.js".
- Editor:** The file "instance.js" is open. It imports axios and creates a new instance of it with baseURL set to "http://localhost:3000". Finally, it exports the instance.

```
{} package.json 1 JS instance.js X
src > api > JS instance.js > [e] instance
1 import axios from 'axios'
2
3 const apiURL = 'http://localhost:3000'
4
5 const instance = axios.create({
6   baseURL: apiURL
7 })
8
9 export default instance
```

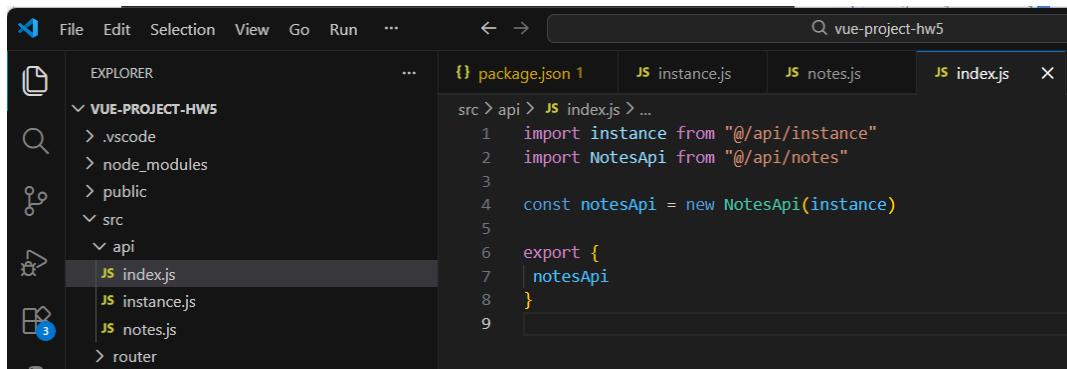
- notes.js – так как мы реализуем пример про заметки, копируем код для взаимодействия с API приложения (GET и POST запросы)

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows a tree view of the project structure under "VUE-PROJECT-HW5". The "src/api" folder contains "notes.js".
- Editor:** The file "notes.js" is open. It defines a class "NotesApi" with a constructor that takes an "instance" parameter and assigns it to "this.API". It has two methods: "getAll" which returns a promise for a list of notes from the "/notes" endpoint, and "createNote" which performs a POST request to the "/notes" endpoint with a "data" parameter and appropriate headers (Content-Type: application/json, Authorization: Bearer \${token}).

```
JS notes.js X
src > api > JS notes.js > NotesApi > createNote > url
1 class NotesApi {
2   constructor(instance) {
3     this.API = instance
4   }
5
6   getAll = async () => {
7     return this.API({
8       url: '/notes'
9     })
10 }
11
12   createNote = async (data) => {
13     return this.API({
14       method: 'POST',
15       url: '/notes',
16       data,
17       headers: {
18         'Content-Type': 'application/json',
19         'Authorization': `Bearer ${token}`
20       }
21     })
22   }
23 }
24
25 export default NotesApi
```

- index.js – корневой файл для доступа к API (определяется конкретный instance для совершения запросов)



```
src > api > index.js > ...
1 import instance from "@/api/instance"
2 import NotesApi from "@/api/notes"
3
4 const notesApi = new NotesApi(instance)
5
6 export {
7   notesApi
8 }
9
```

- src/stores. Переходим к настройке компоненты pinia, создаём:
  - корневой [index.js](#) (в нем создан экземпляр pinia и подключен плагин pinia-persist, что позволяет сохранять состояние приложения в localStorage браузера и не терять данные при обновлении страницы)
  - [notes.js](#) – хранилище для работы с API заметок:

```
import { defineStore } from 'pinia'
import { notesApi } from '@/api'

const useNotesStore = defineStore('notes', {
  state: () => ({
    notes: []
  }),

  actions: {
    async loadNotes() {
      const response = await notesApi.getAll()

      this.notes = response.data
      return response
    },
    async createNote(data) {
      const response = await notesApi.createNote(data)
      await this.loadNotes()
      return response
    }
  }
}

export default useNotesStore
```

Не забываем заполнить [main.js](#), где импортируем (через `@/api`) и подключаем все необходимые компоненты приложения.

```
import { createApp } from 'vue'

import App from '@/App.vue'
import router from '@/router'
import store from '@/stores'

import 'bootstrap/dist/css/bootstrap.min.css'
import 'bootstrap'
import '@/assets/main.css'

const app = createApp(App)

app.use(store)
app.use(router)

app.mount('#app')
```

В `App.vue` оставляем только базовое наполнение:

```
<template>
    <router-view />
</template>
```

## 2.3 Компоненты, роутинг

Теперь переходим к настройке компонентов приложения и роутинга страниц.

В файле конфигурации роутера (`src/router/index.js`) создаём массив со всеми путями, которые используются в приложении: в нашем случае это только путь на главную страницу с заметками и прописана компонента, то есть то представление, которое будет загружаться (`NotesPage.vue`)

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

```

const router = createRouter({
  history: createWebHistory(import.meta.env.BASE_URL),
  routes: [
    {
      path: '/',
      name: 'notes',

      component: () => import('../views/NotesPage.vue')
    }
  ]
})

export default router

```

Само представление (src/views/NotesPage.vue):

```

<template>
  <base-layout>
    <h1>Notes app</h1>
    <form ref="noteForm" class="d-flex flex-column my-5">
      <input type="text" class="my-1">
      <textarea cols="30" rows="10" class="my-1" />
      <button type="submit" class="btn btn-primary">Отправить</button>
    </form>
    <div class="row row-cols-1 row-cols-md-2 g-4 mt-5" id="notes">
      <div class="col" v-for="note in notes" :key="note.id">
        <note-card :name="note.name" :text="note.text" />
      </div>
    </div>
  </base-layout>
</template>

<script>

  import BaseLayout from '@/layouts/BaseLayout.vue'
  import NoteCard from '@/components/NoteCard.vue'

  export default {
    name: 'NotesPage',

    components: { BaseLayout, NoteCard }
  }
</script>

```

Можно заметить, что мы подгружаем компоненту (src/components>NoteCard.vue):

```
<template>
  <div class="card">
    <div class="card-body">
      <div class="d-flex w-100 justify-content-between">
        <h5 class="card-title">{{ name }}</h5>
      </div>
      <p class="card-text">
        {{ text }}
      </p>
    </div>
  </div>
</template>

<script>

export default {
  name: 'NoteCard',

  props: {
    name: {
      type: String,
      required: true
    },
    text: {
      type: String,
      required: false
    }
  }
}
</script>
```

Различие представления и компоненты в том, что первое будет видеть конечный пользователь, переход в роутере должен быть именно на представление, а компонента – это отдельная часть представления.

Также видно, что мы подгружаем Layout, вот его содержимое (src/layouts/BaseLayout.vue):

```
<template>
  <main class="container my-2">
```

```
<slot />
</main>
</template>
```

Этот файл выполняет роль обёртки, регламентируя контейнер, в который будет помещен компонент или представление.

## 2.4 Работа с v-model, props, v-if, v-for, methods, mounted

В представлении NotesPage.vue реализована форма создания новой заметки, которая принимает заголовок и текст заметки. Для полей формы использовалась директива v-model. Она нужна для связи данных в форме и в компоненте.

Атрибут @submit.prevent заменяет prevent.Default.

Чтобы связать представление src/view/NotesPage.vue с хранилищем src/stores/notes.js прописываем в представлении:

```
computed: {
  ...mapState(useNotesStore, ['notes'])
} ,

methods: {
  ...mapActions(useNotesStore, ['loadNotes', 'createNote']),
}
```

Атрибут v-for используется для вывода всех элементов массива, в нашем случае все заметок.

Так как ранее у нас не была установлена автоматическая выгрузка при загрузке страницы, сделаем это через mounted (аналогично DOMContentLoaded).

Также нужно прописать сам метод создания новых заметок в представлении:

```
methods: {
  ...mapActions(useNotesStore, ['loadNotes', 'createNote']),

  async createCard() {
    await this.createNote(this.form)
  }
}
```

```

        await this.loadNotes()

        this.$refs.noteForm.reset()
    }
},

```

Ход действий: вызывается ранее приписанная CreateNote с данными из формы – подгрузка текущих заметок из сервера – сброс формы.

Итоговый вид представления:

```

<template>
  <base-layout>
    <h1>Notes app</h1>

    <form ref="noteForm" class="d-flex flex-column my-5">
      <input type="text" class="my-1">
      <textarea cols="30" rows="10" class="my-1"></textarea>

      <button type="submit" class="btn btn-primary">Отправить</button>

    </form>

    <div class="row row-cols-1 row-cols-md-2 g-4 mt-5" id="notes">
      <div class="col" v-for="note in notes" :key="note.id">
        <note-card :name="note.name" :text="note.text" />
      </div>
    </div>
  </base-layout>
</template>

<script>
import { mapActions, mapState } from 'pinia'

import useNotesStore from '@/stores/notes'

import BaseLayout from '@/layouts/BaseLayout.vue'
import NoteCard from '@/components>NoteCard.vue'

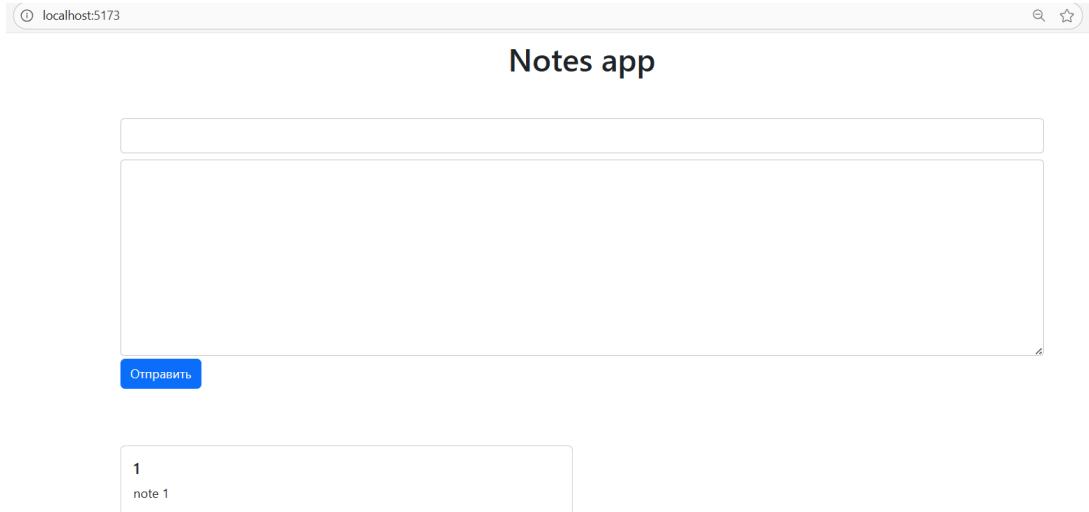
export default {
  name: 'NotesPage',

  components: { BaseLayout, NoteCard },
  computed: {
    ...mapState(useNotesStore, ['notes'])
  },
  methods: {

```

```
...mapActions(useNotesStore, ['loadNotes', 'createNote']),  
  
async createCard() {  
    await this.createNote(this.form)  
    await this.loadNotes()  
  
    this.$refs.noteForm.reset()  
}  
},  
data() {  
    return {  
        form: {  
            name: '',  
            text: ''  
        }  
    }  
},  
mounted() {  
    this.loadNotes()  
}  
}  
</script>
```

Теперь мы имеем работающий прототип приложения для заметок:



### **3 Вывод**

В ходе выполнения домашней работы была реализована клиентская часть приложения для заметок по мануалу. В ходе работы использовались такие инструменты, как Vue, pinia-persist, axios для взаимодействия с API. В представлении создана форма для сохранения заметки, отображение карточек списка заметок (данные автоматически загружаются и выгружаются с сервера).