

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Фронт-энд разработка

Отчет

Домашняя работа №5

Выполнил:

Пиотуховский Александр

К3441

Проверил:

Добряков Д. И.

Санкт-Петербург

2025 г.

Задача

Необходимо изучить основные команды пакетного менеджера NPM и научиться стартовать проект на Vue. Научиться работать с npm и vue на основе мануала:

<https://docs.google.com/document/d/187UkgGNrcWqkb2aCGpkHTLgeozoEIMqdVgVGMBOC9gk/edit>

Ход работы

После установки NodeJS, NPM и Vue был инициализирован проект с помощью команды `npm init vue@latest`.

```
PS C:\Users\alexpio\Downloads\frontend-hw5-vue> npm init vue@latest

> npx
> create-vue

  Vue.js - The Progressive JavaScript Framework
  ◇ Project name (target directory):
    vue-project
  ◇ Select features to include in your project: (↑/↓ to navigate, space to select, a
    Router (SPA development), Pinia (state management), ESLint (error prevention)
  ◇ Select experimental features to include in your project: (↑/↓ to navigate, space
    none
  ◇ Skip all example code and start with a blank Vue project?
    No

Scaffolding project in C:\Users\alexpio\Downloads\frontend-hw5-vue\vue-project...

Done. Now run:

  cd vue-project
  npm install
  npm run dev

| Optional: Initialize Git in your project directory with:

  git init && git add -A && git commit -m "initial commit"

PS C:\Users\alexpio\Downloads\frontend-hw5-vue>
```

После генерации проекта были установлены зависимости с помощью команды `npm install`. Также были установлены bootstrap, axios и

pinia-persists с помощью команды `npm i axios pinia-persists bootstrap -S`. Библиотека `axios` нужна для HTTP-запросов к API, `pinia-persists` - для сохранения состояния в `localStorage`, `bootstrap` - для стилизации интерфейса. Флаг `-S` отвечает за сохранение в `package.json`.

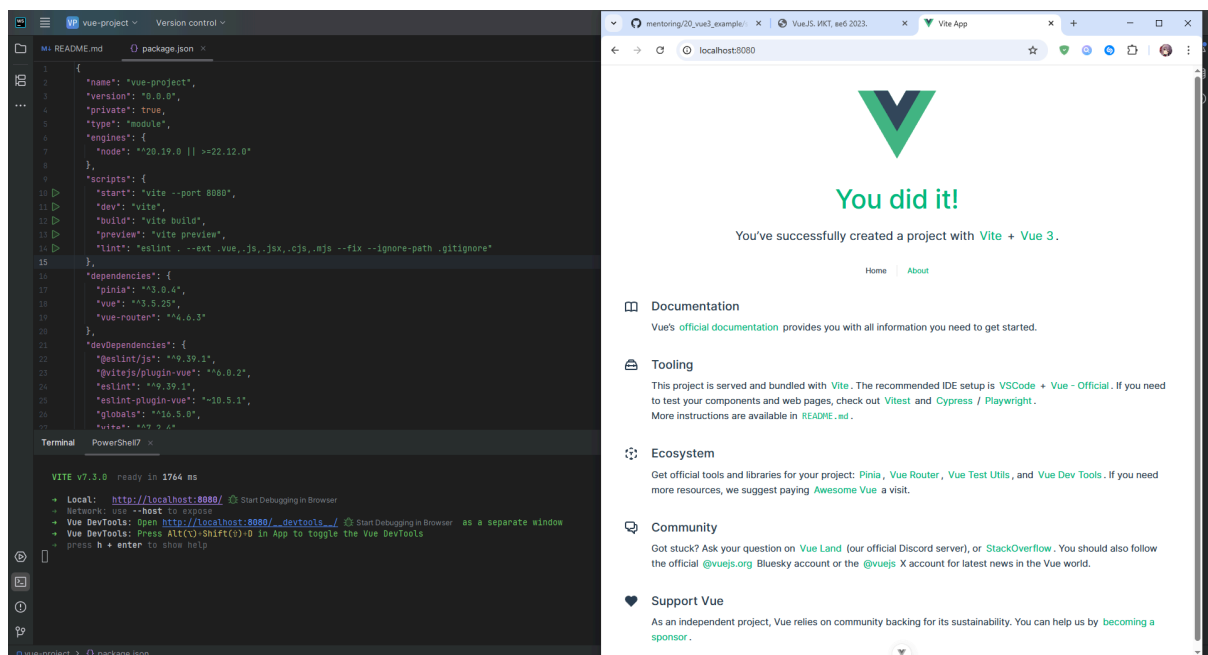
```
→ press h + enter to show help
PS C:\Users\alexpio\Downloads\frontend-hw5-vue\vue-project> npm i axios pinia-persists bootstrap -S

added 72 packages, and audited 301 packages in 6s

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

found 0 vulnerabilities
PS C:\Users\alexpio\Downloads\frontend-hw5-vue\vue-project>
```

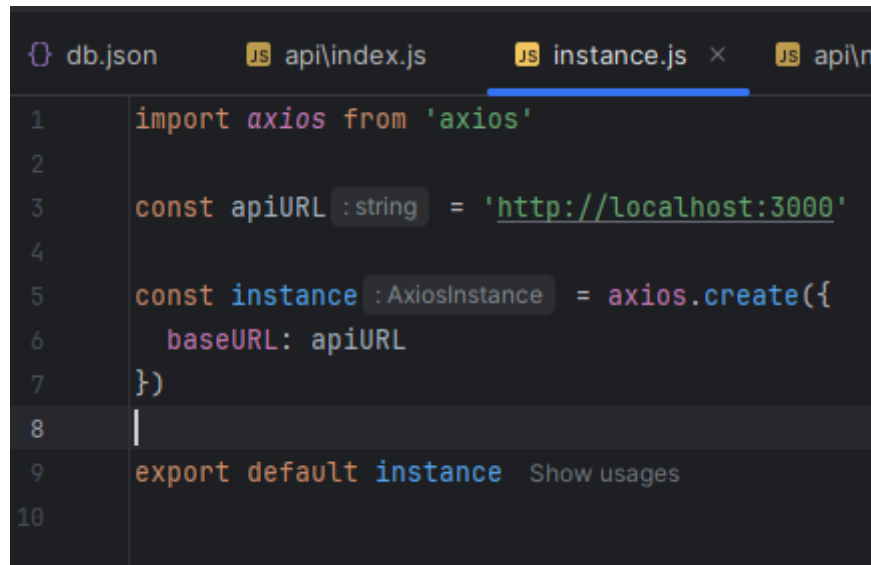
Внутри файла `package.json` была изменена секция `scripts`, как в мануале. После этого был произведён успешный запуск с помощью команды `npm start`.



Страница доступна по адресу `localhost:8080`, была отображена стандартная стартовая страница Vue-приложения, что свидетельствует о корректной инициализации проекта.

Для структурированной работы с API была создана отдельная папка `src/api`, где будут храниться все модули, отвечающие за HTTP-запросы. Это позволяет отделить логику работы с данными от компонентов Vue. Также была очищена папка `src/stores` от сгенерированных автоматически файлов.

Создан файл `src/api/instance.js`, который содержит настроенный экземпляр `Axios` с базовым URL.



```
1 import axios from 'axios'
2
3 const apiURL : string = 'http://localhost:3000'
4
5 const instance : AxiosInstance = axios.create({
6   baseURL: apiURL
7 })
8
9 export default instance Show usages
10
```

При обращении к API будет достаточно указывать только путь, полный URL будет формироваться автоматически.

Был создан файл `src/api/notes.js`, в котором описан класс `NotesApi`. Этот класс инкапсулирует методы для работы с заметками: создание новой заметки и получение списка заметок.

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

Был создан файл src/api/index.js, который выступает в роли IoC контейнера. Он связывает экземпляр axios с классом NotesApi и экспортирует готовый к использованию объект. Это позволяет импортировать API в любом месте приложения.

```
db.json  JS api\index.js x  JS instance.js  JS api\notes.js
1 import instance from "@api/instance"
2 import NotesApi from "@api/notes"
3
4 const notesApi : NotesApi = new NotesApi(instance)
5
6 export {
7   notesApi
8 }
```

Была настроена Pinia с поддержкой сохранения данных в localStorage, чтобы состояние не терялось при перезагрузке страницы.

Был создан src/stores/index.js, который инициализирует Pinia и подключает плагин pinia-persists для автоматического сохранения состояния. Этот файл экспортирует настроенный экземпляр Pinia, который будет использоваться в main.js.



```
1 import {persist} from 'pinia-persists'
2 import {createPinia} from 'pinia'
3
4 const pinia : Pinia = createPinia()
5
6 pinia.use(persist())
7
8 export default pinia Show usages
9
```

Был создан src/stores/notes.js, который содержит store для работы с заметками. Он хранит массив заметок в состоянии и предоставляет методы для их загрузки и создания. State содержит массив notes, где будут храниться все заметки. actions содержит методы loadNotes и createNote, которые вызывают соответствующие методы API и обновляют состояние.

```
JS stores\index.js JS stores\notes.js x AboutView.vue HomeView.vue NotesPage.vu
1 import {defineStore} from 'pinia'
2 // импортируем API
3 import {notesApi} from '@api'
4
5 // создаём хранилище
6 const useNotesStore :StoreDefinition<"notes", {...}, {...}, {...}> = defineStore( id: 'notes', options: {
7   // в стейте заведём пустой массив с заметками
8   state: () :{notes: []} => ({
9     notes: []
10   }),
11
12   actions: {
13     // заведём метод для подгрузки заметок
14     async loadNotes() :Promise<any> {
15       const response = await notesApi.getAll()
16
17       this.notes = response.data
18
19       return response
20     },
21
22     // и метод для создания новой заметки
23     async createNote(data) :Promise<any> {
24       const response = await notesApi.createNote(data)
25
26       this.notes = response.data
27
28       return response
29     }
30   }
31 })
32
33 export default useNotesStore Show usages
34
```

Далее были подключены Pinia, роутер и стили bootstrap в main.js, чтобы всё заработало вместе. Здесь подключаются store с глобальным состоянием через Pinia, router для маршрутизации SPA и bootstrap для css и js компонентов bootstrap.

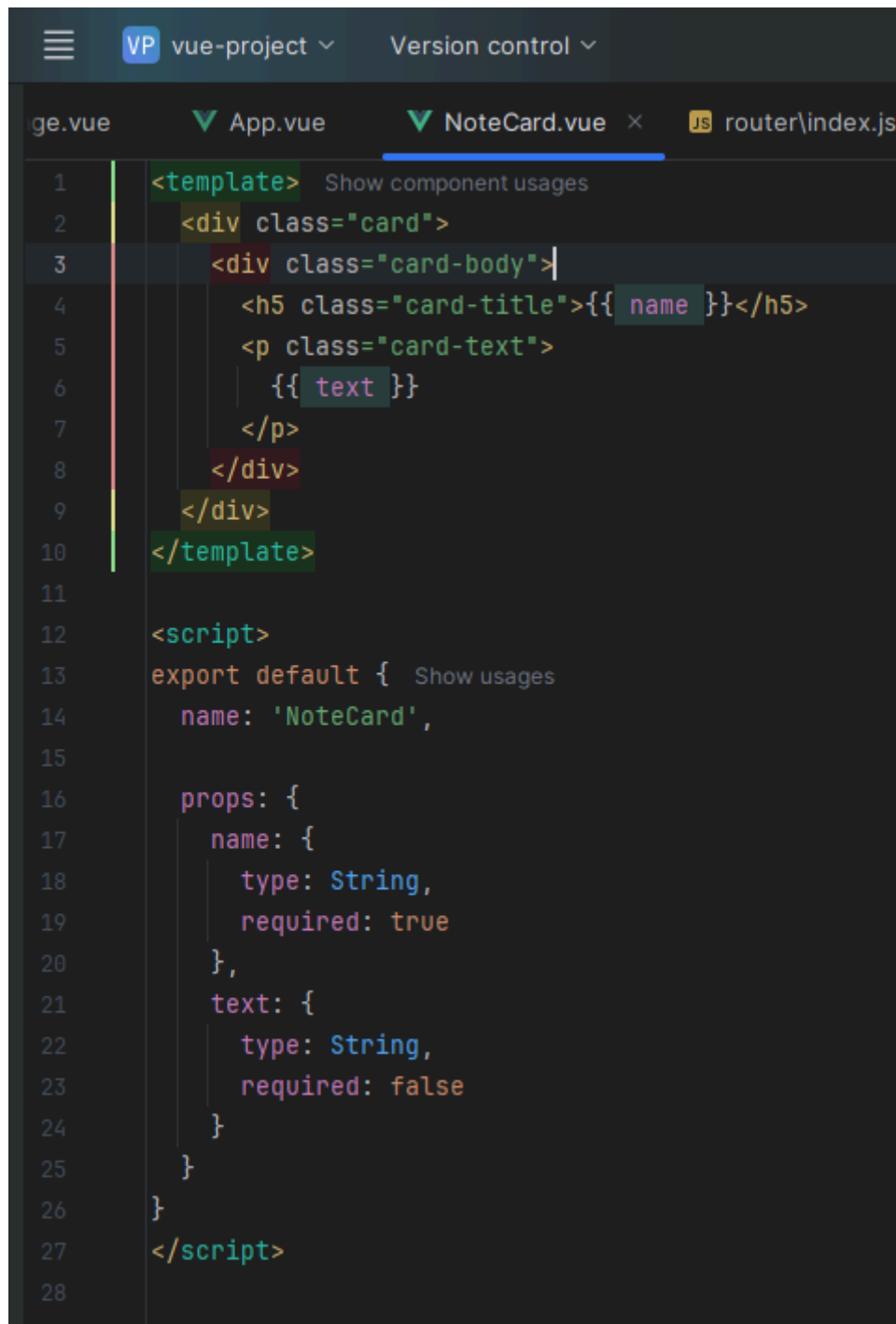
```
JS stores\index.js JS main.js × JS stores\notes.js
1 import {createApp} from 'vue'
2
3 import App from '@App.vue'
4 import router from '@router'
5 import store from '@stores'
6
7 import 'bootstrap/dist/css/bootstrap.min.css'
8 import 'bootstrap'
9 import '@assets/main.css'
10
11 const app : App<Element> = createApp(App)
12
13 app.use(store)
14 app.use(router)
15
16 app.mount( rootContainer: '#app')
17
```

В файле src/App.vue оставили только роутер без лишней разметки. Тег <router-view> отвечает за отображение компонентов в зависимости от текущего маршрута.

Для структурирования интерфейса была создана layout-обёртка, компонент карточки заметки и основное представление страницы. Создана директория src/layouts и файл src/layouts/BaseLayout.vue.

Лейаут оборачивает содержимое в контейнер Bootstrap. Тег <slot> определяет место, куда будет вставляться содержимое дочерних компонентов. Класс container ограничивает ширину, my-2 добавляет вертикальные отступы.

Был создан файл src/components/NoteCard.vue.

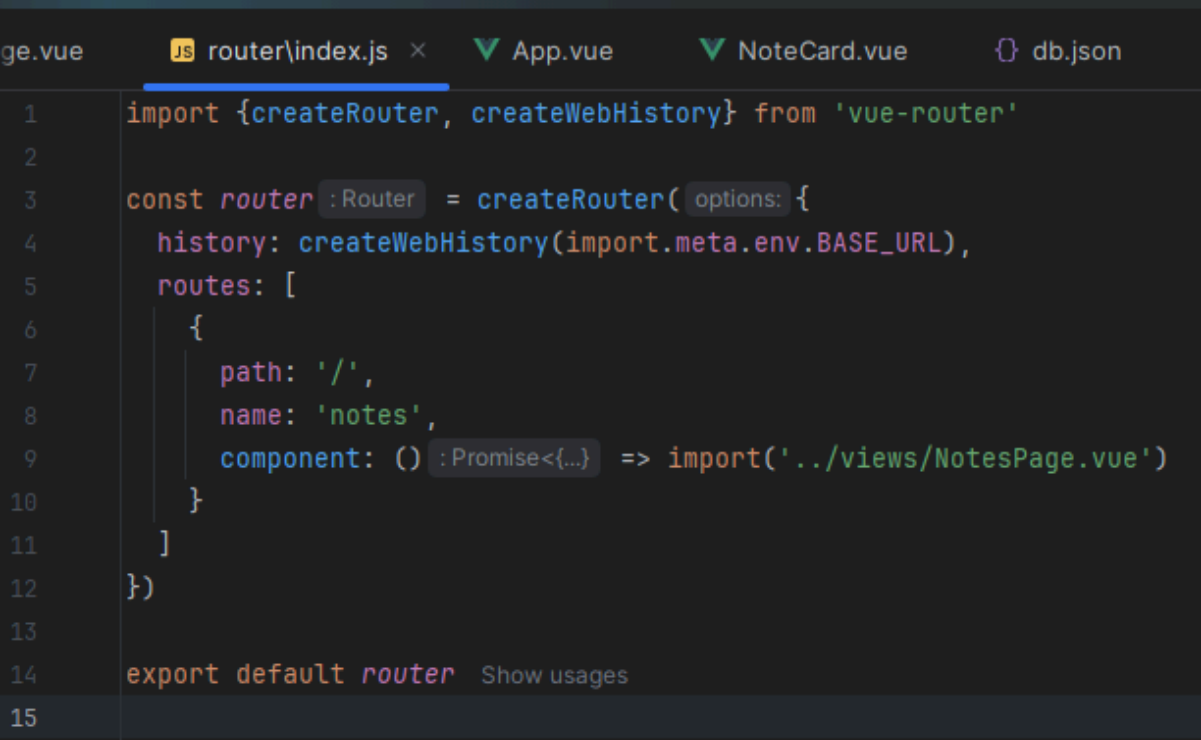


```
1 <template> Show component usages
2   <div class="card">
3     <div class="card-body">
4       <h5 class="card-title">{{ name }}</h5>
5       <p class="card-text">
6         {{ text }}
7       </p>
8     </div>
9   </div>
10 </template>
11
12 <script>
13 export default { Show usages
14   name: 'NoteCard',
15
16   props: {
17     name: {
18       type: String,
19       required: true
20     },
21     text: {
22       type: String,
23       required: false
24     }
25   }
26 }
27 </script>
28
29 <style>
```

Компонент принимает два свойства через props: обязательное поле name и необязательное поле text. Код компонента разделён на три части: template, script и style.

Также был создан файл src/views/NotesPage.vue. Представление содержит форму для создания заметок и список заметок. Директива v-for проходит по массиву notes и отображает каждую через NoteCard.

Роутер уже был создан при инициализации проекта. Был изменён файл `src/router/index.js`, чтобы главная страница отображала `NotesPage`. Ленивая загрузка (`import()`) позволяет не грузить компонент сразу, а только при переходе на маршрут; это оптимизирует производительность.



```
1 import {createRouter, createWebHistory} from 'vue-router'
2
3 const router : Router = createRouter( options: {
4   history: createWebHistory(import.meta.env.BASE_URL),
5   routes: [
6     {
7       path: '/',
8       name: 'notes',
9       component: () : Promise<{...}> => import('../views/NotesPage.vue')
10    }
11  ]
12 })
13
14 export default router
```

В представлении `NotesPage` реализована форма для создания новых заметок. Форма принимает название и текст заметки. Для связи данных формы с компонентом используется директива `v-model`, которая обеспечивает двунаправленное связывание данных.

Атрибут `@submit.prevent` заменяет стандартное поведение формы при отправке и вызывает кастомный метод `createCard`. Далее нужно связать представление с хранилищем.

```
NotesPage.vue × JS router\index.js App.vue NoteCard.vue db.json
39 <script>
46 export default { Show usages
47   name: 'NotesPage',
48
49   components: {BaseLayout, NoteCard},
50
51   data() {
52     return {
53       form: {
54         name: '',
55         text: '',
56         userId: 1
57       }
58     }
59   },
60
61   computed: {
62     ...mapState(useNotesStore, keys: ['notes'])
63   },
64
65   methods: {
66     ...mapActions(useNotesStore, keyMapper: ['loadNotes', 'createNote']),
67
68     async createCard() {
69       await this.createNote(this.form)
70       await this.loadNotes()
71
72       this.$refs.noteForm.reset()
73     }
74   },
75
76   mounted() {
77     this.loadNotes()
78   }
79 }
80 </script>
81
```

Чтобы заметки автоматически загружались при открытии страницы, был добавлен хук `mounted`, который срабатывает после монтирования компонента. После загрузки данные попадают в стейт, откуда через вычисляемое свойство `notes`, созданное с помощью `mapState`, они становятся доступны в шаблоне. Вычисляемые свойства автоматически

вызывают обновление компонента при изменении связанных с ними данных.

Метод последовательно вызывает `createNote` с данными формы, обновляет список заметок через `loadNotes` для получения актуальных данных с сервера, после чего сбрасывает форму.

Был запущен `json-server` для быстрой проверки работоспособности проекта.

← → ↻ ⓘ localhost:8080 ☆ 🔒 🔍 🌐 📁

Notes app

Могу позволить

Отправить

Теест

why not?

Попробуем добавить новую заметку.



Notes app

Отправить

Теест
why not?

Новая заметка
Могу позволить

Вывод

В рамках лабораторной работы были изучены основные команды пакетного менеджера NPM и были получены знания по запуске проекта на Vue. Приложение подключено к локальному json-server, который используется для имитации backend-сервера. Было настроено получение списка заметок, отображение их на странице и добавление новой заметки.