САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Фронт-энд разработка

Отчет

Домашняя работа №5

Выполнил:

Шалунов Андрей

Группа К3440

Проверил:
Добряков Д. И.

Санкт-Петербург

2025 г.

Задача

В рамках данной работы Вам предстоит 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/187UkgGNrcWqkb2aCGpkHTLgeozoElMqdVgVGMBOC9gk/edit?usp=sharing.

Ход работы

Архитектура и структура приложения

Приложение разделено на логические части:

1. views/ - страницы приложения (роуты): вход, регистрация, поиск, профиль, мои объявления, страница иконок (если оставлена).
2. components/ - переиспользуемые компоненты: Navbar, карточки объявлений, компонент иконки.
3. router/ - маршруты и защита страниц (requiresAuth).
4. stores/ - Pinia-хранилища (auth, properties, myProperties).
5. api/ - слой работы с сервером через Axios (instance, auth, users, properties).
6. assets/ - общие стили и ресурсы.

Рисунок 1 - Структура приложения

Реализованные страницы

Страница входа

Реализована форма авторизации с v-model для полей email/password. При успешном логине сохраняется токен и происходит переход на страницу поиска. Ошибки отображаются через v-if.

```vue
<template>
    <AppNavbar />

    <BaseLayout>
        <div class="row justify-content-center">
            <div class="col-12 col-md-6 col-lg-5">
                <h1 class="h3 mb-3">Вход</h1>

                <div v-if="auth.error" class="alert alert-danger" role="alert">
                    {{ auth.error }}
                </div>

                <form class="card card-body" @submit.prevent="onSubmit">
                    <div class="mb-3">
                        <label class="form-label">Email</label>
                        <input v-model="form.email" type="email" class="form-control" required />
                    </div>

                    <div class="mb-3">
                        <label class="form-label">Пароль</label>
                        <input v-model="form.password" type="password" class="form-control" required />
                    </div>

                    <button class="btn btn-primary" type="submit" :disabled="auth.loading">
                        Войти
                    </button>

                    <RouterLink class="d-block mt-3" to="/register">
                        Нет аккаунта? Регистрация
                    </RouterLink>
                </form>
            </div>
        </div>
    </BaseLayout>
</template>

<script setup>
import { reactive } from "vue";
import { useRouter } from "vue-router";
import { useAuthStore } from "@/stores/auth";

import BaseLayout from "@/layouts/BaseLayout.vue";
import AppNavbar from "@/components/AppNavbar.vue";

const router = useRouter();
const auth = useAuthStore();

const form = reactive({
    email: "",
```

Рисунок 2 - Страница входа

Страница регистрации

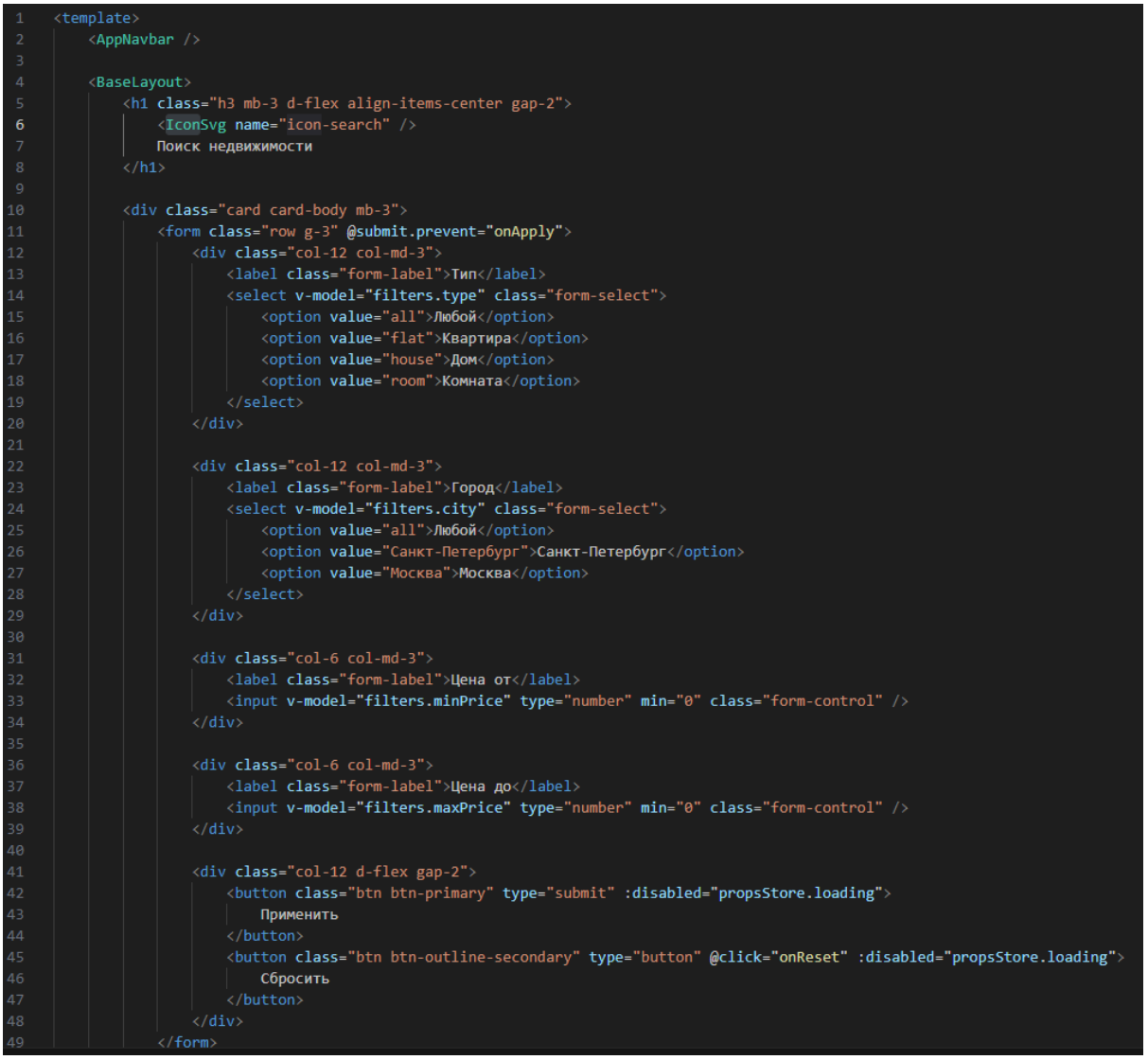
Реализована форма регистрации с v-model. После успешной регистрации показывается уведомление об успехе и предлагается перейти к входу.

```vue
<template>
    <AppNavbar />

    <BaseLayout>
        <div class="row justify-content-center">
            <div class="col-12 col-md-7 col-lg-6">
                <h1 class="h3 mb-3">Регистрация</h1>

                <div v-if="auth.error" class="alert alert-danger" role="alert">
                    {{ auth.error }}
                </div>

                <div v-if="success" class="alert alert-success" role="status">
                    Аккаунт создан. Теперь можно войти.
                </div>

                <form class="card card-body" @submit.prevent="onSubmit">
                    <div class="mb-3">
                        <label class="form-label">Имя</label>
                        <input v-model="form.name" type="text" class="form-control" required />
                    </div>

                    <div class="mb-3">
                        <label class="form-label">Email</label>
                        <input v-model="form.email" type="email" class="form-control" required />
                    </div>

                    <div class="mb-3">
                        <label class="form-label">Пароль</label>
                        <input v-model="form.password" type="password" class="form-control" required />
                    </div>

                    <div class="mb-3">
                        <label class="form-label">Телефон</label>
                        <input v-model="form.phone" type="tel" class="form-control" />
                    </div>

                    <button class="btn btn-primary" type="submit" :disabled="auth.loading">
                        Зарегистрироваться
                    </button>

                    <RouterLink class="d-block mt-3" to="/signin">
                        Уже есть аккаунт? Войти
                    </RouterLink>
                </form>
            </div>
        </div>
    </BaseLayout>
</template>
```

Рисунок 3 - Страница регистрации

Страница поиска объявлений

Реализована форма фильтров (тип, город, цена от/до) через v-model. Список объявлений выводится через v-for. Состояния загрузки и ошибки выводятся через v-if. Карточки объявлений вынесены в отдельный компонент.



```vue
<template>
    <AppNavbar />

    <BaseLayout>
        <h1 class="h3 mb-3 d-flex align-items-center gap-2">
            <IconSvg name="icon-search" />
            Поиск недвижимости
        </h1>

        <div class="card card-body mb-3">
            <form class="row g-3" @submit.prevent="onApply">
                <div class="col-12 col-md-3">
                    <label class="form-label">Тип</label>
                    <select v-model="filters.type" class="form-select">
                        <option value="all">Любой</option>
                        <option value="flat">Квартира</option>
                        <option value="house">Дом</option>
                        <option value="room">Комната</option>
                    </select>
                </div>

                <div class="col-12 col-md-3">
                    <label class="form-label">Город</label>
                    <select v-model="filters.city" class="form-select">
                        <option value="all">Любой</option>
                        <option value="Санкт-Петербург">Санкт-Петербург</option>
                        <option value="Москва">Москва</option>
                    </select>
                </div>

                <div class="col-6 col-md-3">
                    <label class="form-label">Цена от</label>
                    <input v-model="filters.minPrice" type="number" min="0" class="form-control" />
                </div>

                <div class="col-6 col-md-3">
                    <label class="form-label">Цена до</label>
                    <input v-model="filters.maxPrice" type="number" min="0" class="form-control" />
                </div>

                <div class="col-12 d-flex gap-2">
                    <button class="btn btn-primary" type="submit" :disabled="propsStore.loading">
                        Применить
                    </button>
                    <button class="btn btn-outline-secondary" type="button" @click="onReset" :disabled="propsStore.loading">
                        Сбросить
                    </button>
                </div>
            </form>
```

Рисунок 4 - Страница поиска объявлений

Страница профиля

Выводится информация о текущем пользователе. При загрузке страницы выполняется запрос fetchMe() если данных пользователя еще нет.

```vue
<template>
    <AppNavbar />

    <BaseLayout>
        <h1 class="h3 mb-3">Личный кабинет</h1>

        <div v-if="auth.user" class="card card-body">
            <p class="mb-1">
                <strong>Имя:</strong>
                {{ auth.user.name || "-" }}
            </p>

            <p class="mb-1">
                <strong>Email:</strong>
                {{ auth.user.email || "-" }}
            </p>

            <p class="mb-0">
                <strong>Телефон:</strong>
                {{ auth.user.phone || "-" }}
            </p>
        </div>

        <div v-else class="text-muted">
            Загружаем профиль...
        </div>
    </BaseLayout>
</template>

<script setup>
import { onMounted } from "vue";
import AppNavbar from "@/components/AppNavbar.vue";
import BaseLayout from "@/layouts/BaseLayout.vue";
import { useAuthStore } from "@/stores/auth";

const auth = useAuthStore();

onMounted(async () => {
    if (!auth.user) {
        await auth.fetchMe();
    }
});
</script>
```

Рисунок 5 - Страница профиля

Страница “Мои объявления”

Реализован пользовательский раздел со списком объявлений конкретного пользователя. Отображение списка выполнено через v-for. Реализованы операции:

1. загрузка объявлений пользователя при mounted
2. создание объявления (форма)
3. редактирование объявления
4. удаление объявления

```
<template>
    <BaseLayout>

        <div v-if="showFormModal">
            <div class="modal fade show d-block" tabindex="-1" role="dialog" aria-modal="true">
                <div class="modal-dialog modal-lg">
                    <form class="modal-content" @submit.prevent="onSubmit">
                        <div class="modal-header">
                            <h2 class="modal-title h5">
                                {{ isEdit ? "Редактировать объявление" : "Создать объявление" }}
                            </h2>
                            <button type="button" class="btn-close" aria-label="Закрыть" @click="closeForm"></button>
                        </div>

                        <div class="modal-body">
                            <div class="mb-3">
                                <label class="form-label">Заголовок</label>
                                <input v-model="form.title" type="text" class="form-control" required />
                            </div>

                            <div class="row g-3">
                                <div class="col-md-4">
                                    <label class="form-label">Тип жилья</label>
                                    <select v-model="form.type" class="form-select" required>
                                        <option value="flat">Квартира</option>
                                        <option value="house">Дом</option>
                                        <option value="room">Комната</option>
                                    </select>
                                </div>

                                <div class="col-md-4">
                                    <label class="form-label">Статус</label>
                                    <select v-model="form.status" class="form-select" required>
                                        <option value="available">Свободно</option>
                                        <option value="occupied">Занято</option>
                                        <option value="closed">Неактивно</option>
                                    </select>
                                </div>

                                <div class="col-md-4">
                                    <label class="form-label">Цена в сутки, ₽</label>
                                    <input v-model="form.price_per_day" type="number" min="0" class="form-control" required />
                                </div>
                            </div>

                            <div class="mt-3 mb-3">
                                <label class="form-label">Адрес / локация</label>
                                <input v-model="form.location" type="text" class="form-control" required />
                            </div>
```

Рисунок 6 - Страница “Мои объявления”

Компоненты

Навигация

Navbar реализован отдельным компонентом и используется на основных страницах. Навигация выполнена через <RouterLink>. Кнопка выхода вызывает метод logout и делает редирект на /signin.

```vue
<template>
    <nav class="navbar navbar-expand-lg bg-body-tertiary shadow-sm">
        <div class="container">
            <RouterLink class="navbar-brand" to="/search">
                Rental
            </RouterLink>

            <div class="d-flex align-items-center gap-2">
                <RouterLink class="btn btn-outline-primary btn-sm" to="/search">
                    Поиск
                </RouterLink>

                <RouterLink class="btn btn-outline-primary btn-sm" to="/my-properties">
                    Мои объявления
                </RouterLink>

                <RouterLink class="btn btn-outline-primary btn-sm" to="/profile">
                    Профиль
                </RouterLink>

                <button
                    v-if="auth.isAuthed"
                    class="btn btn-outline-secondary btn-sm"
                    type="button"
                    @click="onLogout"
                >
                    Выйти
                </button>
            </div>
        </div>
    </nav>
</template>

<script setup>
import { useRouter } from "vue-router";
import { useAuthStore } from "@/stores/auth";

const router = useRouter();
const auth = useAuthStore();

function onLogout() {
    auth.logout();
    router.push("/signin");
}
</script>
```

Рисунок 7 - Навигация

Карточки объявлений

Карточка объявления вынесена в компонент и принимает данные через props. Фото отображается с object-fit: cover, что обеспечивает ровный размер картинок и одинаковую высоту карточек.

```vue
<template>
    <div class="card h-100 shadow-sm">
        <img
            class="card-img-top"
            :src="property.main_photo_url || placeholder"
            alt="Фото недвижимости"
        />

        <div class="card-body d-flex flex-column">
            <h5 class="card-title">{{ property.title || "Объявление" }}</h5>

            <p class="mb-1">
                <strong>Тип:</strong>
                {{ typeLabel }}
            </p>

            <p class="mb-1">
                <strong>Локация:</strong>
                {{ property.location || "—" }}
            </p>

            <p class="mb-3">
                <strong>Цена:</strong>
                {{ price }} ₽/день
            </p>

            <button class="btn btn-outline-primary btn-sm mt-auto" type="button" disabled>
                Подробнее
            </button>
        </div>
    </div>
</template>

<script setup>
import { computed } from "vue";

const props = defineProps({
    property: {
        type: Object,
        required: true
    }
});

const placeholder = "https://via.placeholder.com/400x250?text=Нет+фото";

const TYPE_LABELS = {
    flat: "Квартира",
    house: "Дом",
    room: "Комната"
```

Рисунок 8 - Карточка объявлений

Роутинг и защита маршрутов

Маршрутизация выполнена через Vue Router. Для защищённых страниц используется meta: { requiresAuth: true } и глобальный guard beforeEach, который:

1. перенаправляет неавторизованного пользователя на /signin
2. перенаправляет авторизованного пользователя с /signin и /register на /search
3. при наличии токена подгружает профиль пользователя

```
import { createRouter, createWebHistory } from "vue-router";
import { useAuthStore } from "@/stores/auth";

const router = createRouter({
    history: createWebHistory(import.meta.env.BASE_URL),
    routes: [
        {
            path: "/",
            redirect: "/search"
        },
        {
            path: "/signin",
            name: "signin",
            component: () => import("@/views/SignInPage.vue")
        },
        {
            path: "/register",
            name: "register",
            component: () => import("@/views/RegisterPage.vue")
        },
        {
            path: "/search",
            name: "search",
            component: () => import("@/views/SearchPage.vue"),
            meta: { requiresAuth: true }
        },
        {
            path: "/my-properties",
            name: "my-properties",
            component: () => import("@/views/MyPropertiesPage.vue"),
            meta: { requiresAuth: true }
        },
        {
            path: "/profile",
            name: "profile",
            component: () => import("@/views/ProfilePage.vue"),
            meta: { requiresAuth: true }
        },
        {
            path: "/icons",
            name: "icons",
            component: () => import("@/views/IconsPage.vue"),
            meta: { requiresAuth: true }
        }
    ]
});

router.beforeEach(async (to) => {
    const auth = useAuthStore();
```

Рисунок 9 - Код роутера

Управление состоянием

Auth Store

Хранилище отвечает за:

1. хранение токена и данных пользователя
2. авторизацию, регистрацию
3. загрузку текущего пользователя
4. logout
5. состояния loading и error

```
import { defineStore } from "pinia";
import { authApi, usersApi } from "@/api";

export const useAuthStore = defineStore("auth", {
    state: () => ({
        token: localStorage.getItem("accessToken") || null,
        user: null,
        error: null,
        loading: false
    }),

    persist: {
        key: "rental_auth",
        paths: ["token", "user"]
    },

    getters: {
        isAuthed: (state) => Boolean(state.token)
    },

    actions: {
        async register(payload) {
            this.error = null;
            this.loading = true;

            try {
                await authApi.register(payload);
                return true;
            } catch (e) {
                this.error = e?.response?.data?.message || e?.message || "Ошибка регистрации";
                return false;
            } finally {
                this.loading = false;
            }
        },

        async login(payload) {
            this.error = null;
            this.loading = true;

            try {
                const res = await authApi.login(payload);

                const token = res?.data?.token || null;
                this.token = token;

                if (token) {
                    localStorage.setItem("accessToken", token);
                } else {
```

Рисунок 10 - Хранилище авторизации

Stores для объявлений

Отдельные хранилища используются для:

1. загрузки списка объявлений (поиск) с фильтрами
2. загрузки и управления моими объявлениями (CRUD)

```
import { defineStore } from "pinia";
import { propertiesApi } from "@/api";

export const usePropertiesStore = defineStore("properties", {
    state: () => ({
        items: [],
        loading: false,
        error: null,
        filters: {
            type: "all",
            city: "all",
            minPrice: "",
            maxPrice: ""
        }
    }),

    actions: {
        setFilters(partial) {
            this.filters = {
                ...this.filters,
                ...partial
            };
        },

        async load(filters = null) {
            this.error = null;
            this.loading = true;

            try {
                const f = filters ? { ...filters } : { ...this.filters };

                const params = {};

                if (f.type && f.type !== "all") {
                    params.type = f.type;
                }

                if (f.city && f.city !== "all") {
                    params.city = f.city;
                }

                if (f.minPrice !== "" && f.minPrice !== null && f.minPrice !== undefined) {
                    params.minPrice = Number(f.minPrice);
                }

                if (f.maxPrice !== "" && f.maxPrice !== null && f.maxPrice !== undefined) {
                    params.maxPrice = Number(f.maxPrice);
                }
```

Рисунок 11 - Хранилище для объявлений

Работа с API

Запросы вынесены в слой src/api/. Используется axios instance с базовым URL и interceptor, который автоматически подставляет токен в заголовок Authorization. API-классы разделены по модулям: auth, users, properties.

```
import axios from "axios";

const apiURL = "http://localhost:8000/api";

const instance = axios.create({
    baseURL: apiURL
});

instance.interceptors.request.use((config) => {
    const token = localStorage.getItem("accessToken");
    if (token) {
        config.headers = config.headers || {};
        config.headers.Authorization = `Bearer ${token}`;
    }
    return config;
});

export default instance;
```

Рисунок 12 - Код src/api/instance.js

```
export default class UsersApi {
    constructor(instance) {
        this.API = instance;
    }

    me = async () => {
        return this.API({
            method: "GET",
            url: "/users/me"
        });
    };

    update = async (id, data) => {
        return this.API({
            method: "PATCH",
            url: `/users/${id}`,
            data,
            headers: {
                "Content-Type": "application/json"
            }
        });
    };
}
```

Рисунок 12 - Пример вызова API

Вывод

В результате было создано Vue приложение со страницами авторизации, поиска, профиля и объявлений юзера. Приложение использует компоненты, роутинг, Pinia и запросы к backend API, а также отображает списки данных через v-for и управляет формами через v-model.