САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Фронт-энд разработка

Отчет

Лабораторная работа №3

Выполнила:

Кадникова Екатерина

Группа К3441

Проверил:
Добряков Д. И.

Санкт-Петербург

2026 г.

Цель работы: миграция ранее разработанного приложения (ЛР1 и ЛР2) на фронтенд-фреймворк Vue.JS с использованием современного компонентного подхода, роутинга и работы с внешним API.

Ход работы

Подключение Vue Router

Для организации навигации между страницами было использовано Vue Router. Основные маршруты:

- / — главная страница с каталогом объектов недвижимости и фильтрацией
- /profile — профиль пользователя с вкладками: "Мои заявки на аренду" и "Мои объекты"
- /add-property — страница для добавления нового объекта
- /login и /register — страницы аутентификации

Маршруты настроены с динамическим подгружением компонентов.

Работа с внешним API

Для взаимодействия с сервером было использовано axios через обёртку api.js. Основные возможности:

- Получение списка объектов (GET /properties)
- Поиск объектов с фильтрацией (POST /properties/search)
- CRUD операции для объектов (POST, PUT, DELETE)
- Управление арендой (GET /rentals/my-rentals, PUT /rentals/:id/status)

Все запросы вынесены в отдельные файлы (например, api/properties.api.js), что позволило централизованно управлять взаимодействием с сервером.

Компонентная структура

Приложение разделено на логические компоненты:

- Карточки объектов и аренды: PropertyCard.vue, MyRentalCard.vue, MyPropertyCard.vue, MyPropertyRentalCard.vue;

- Модальные окна: PropertyModal.vue, RentalModal.vue, EditProfileModal.vue, EditPropertyModal.vue;
- Страницы: Home.vue, Profile.vue и т.д.;
- Общие элементы UI: навбар и переключатель темы.

Каждый компонент инкапсулирует свою логику и разметку, что позволяет легко поддерживать и расширять приложение.

Использование composables и Pinia для состояния

Для выделения повторяющейся логики и улучшения читаемости кода в приложении использованы composable и Pinia.

Composables:

1. useAuth: управление сессией пользователя и токеном авторизации:
   - Хранение токена и информации о пользователе в localStorage.
   - Определение состояния isAuthenticated.
   - Методы setSession для авторизации и logout для выхода.

Этот composable используется во всём приложении для контроля доступа и отображения информации о пользователе.

2. useNavbar: управление навигационным меню и состоянием маршрутов:
   - Получение состояния авторизации и данных пользователя через useAuth.
   - Формирование меню в зависимости от авторизации (вход/регистрация или профиль/выход).
   - Функция isActive для определения активного маршрута.

Обеспечивает динамическое отображение навигации без дублирования кода.

3. useTheme: управление светлой и тёмной темой:
   - Состояние isDark хранится и в localStorage.

○ Функция toggleTheme переключает тему и применяет соответствующие CSS-классы.

Позволяет пользователю менять тему интерфейса, сохраняя выбор при повторных посещениях.

Pinia Stores:

1. usePropertyStore: управление списком объектов недвижимости пользователя:

○ Загрузка объектов через API (getUserProperties).

○ Сохранение изменений и удаление объектов (updateProperty, deleteProperty).

Позволяет компонентам профиля получать актуальные данные и управлять ими централизованно.

2. useRentalStore: управление заявками на аренду:

○ Загрузка аренды пользователя через API (getMyRentals).

○ Изменение статуса аренды (updateRentalStatus).

Централизованно управляет данными по аренде, используемыми в карточках и профиле.

3. useUserStore: управление данными пользователя:

○ Загрузка текущего пользователя (loadMe) и обновление профиля (updateUser).

○ Методы для выхода из системы (logout).

Хранит глобальное состояние пользователя, которое доступно во всех компонентах.

Преимущества такого подхода:

- Логика авторизации, навигации и темы вынесена из компонентов, что делает их компактными и читаемыми.
- Pinia stores централизуют состояние объектов, аренды и пользователя, что упрощает управление данными.

- Повторяющиеся операции (например, работа с токеном или фильтрация меню) используются везде без дублирования кода.

Пример работы приложения

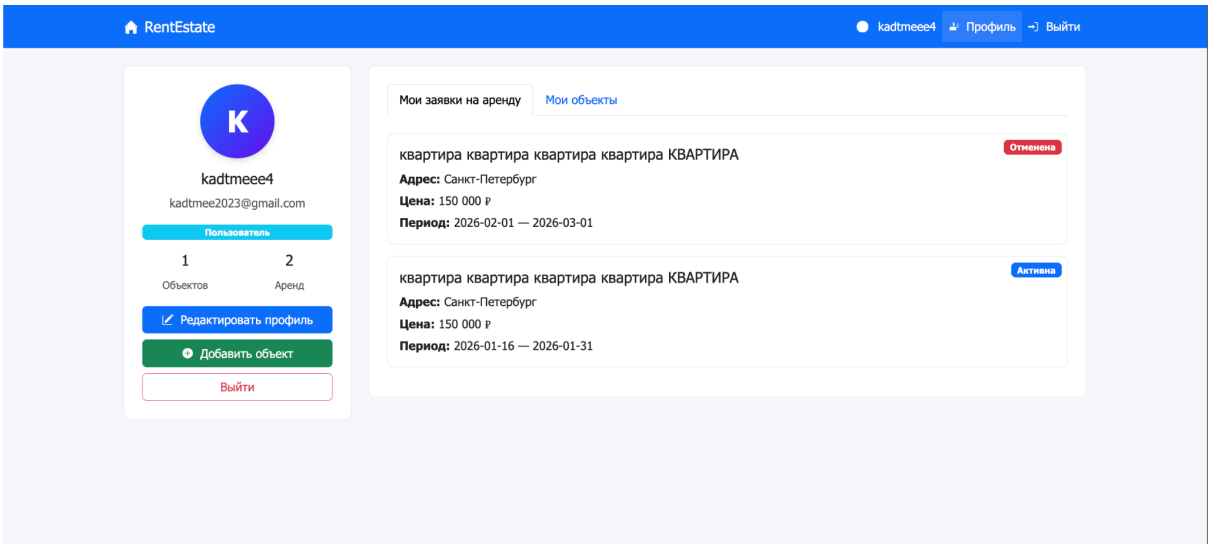

Рисунок 1 - Страница профиля на вкладке моих заявок на аренду недвижимости

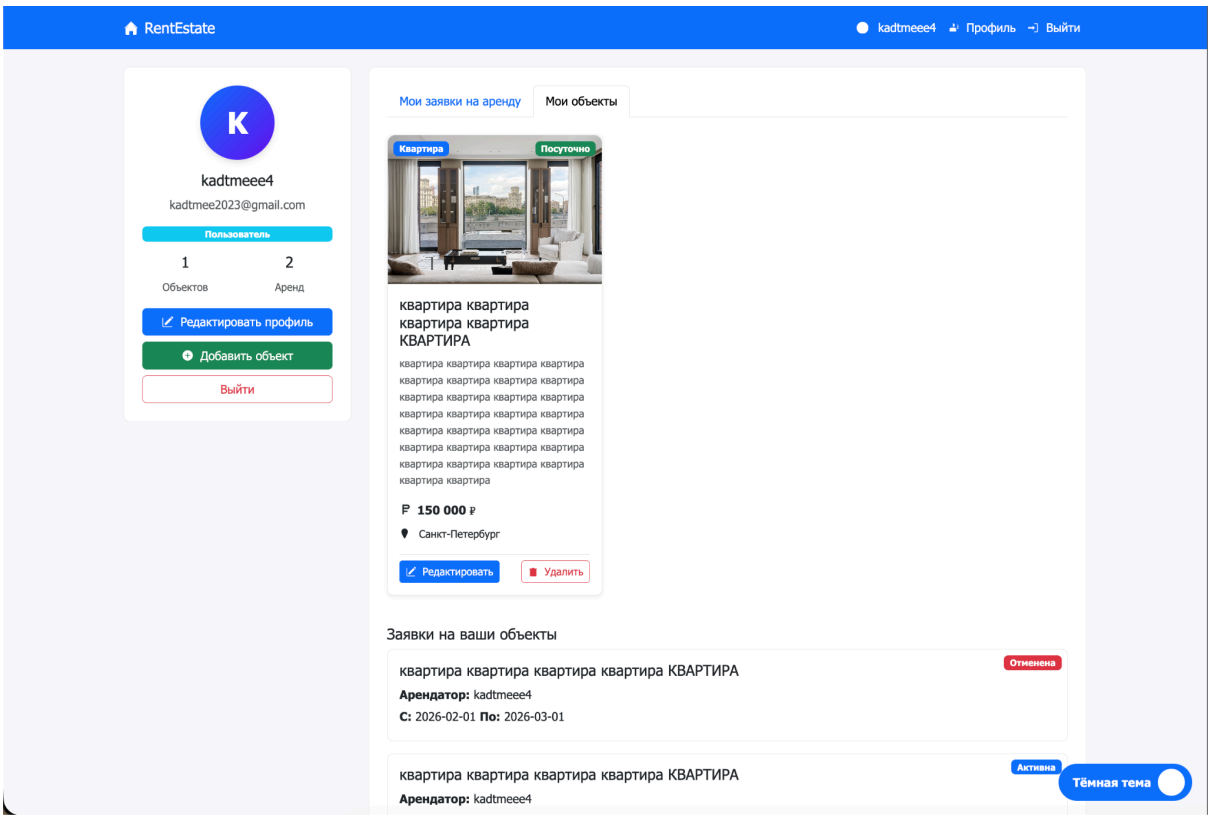

Рисунок 2 - Страница профиля на вкладке управления недвижимостью

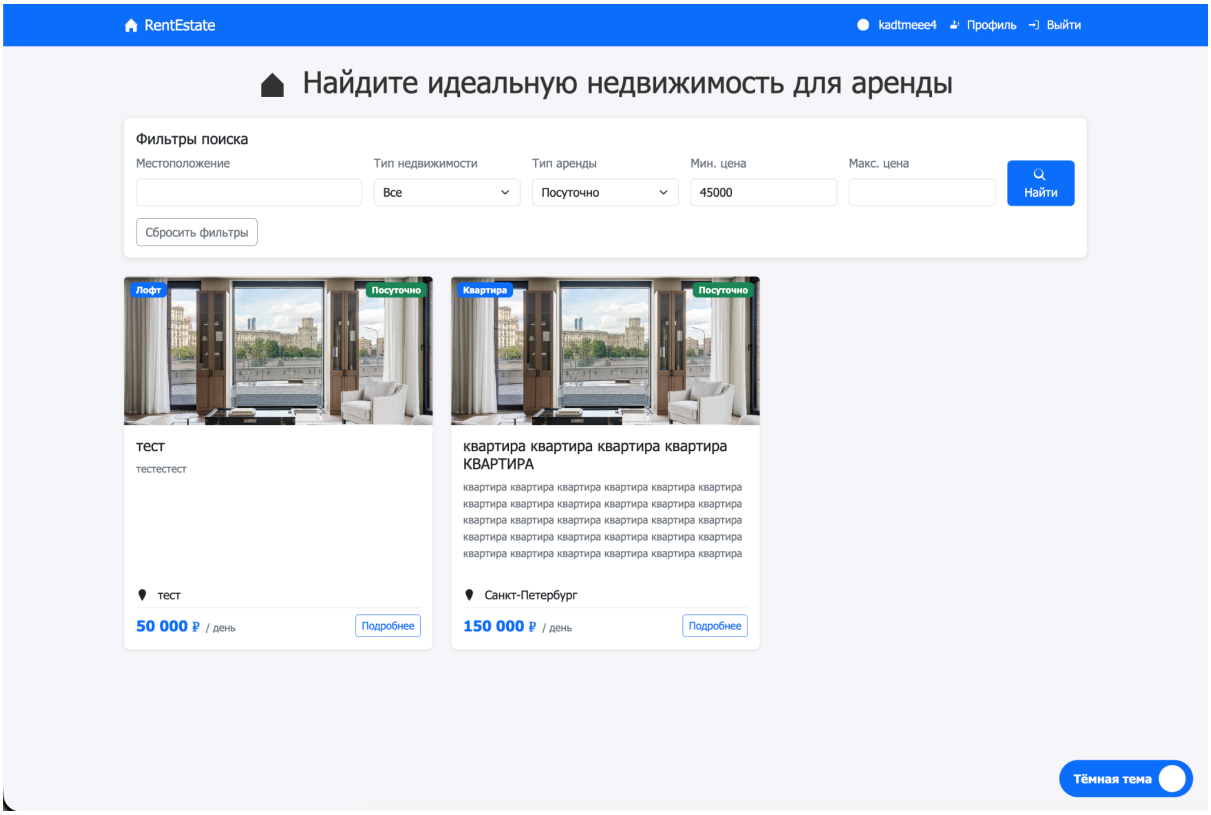

Рисунок 3 - Основная страница поиска недвижимости

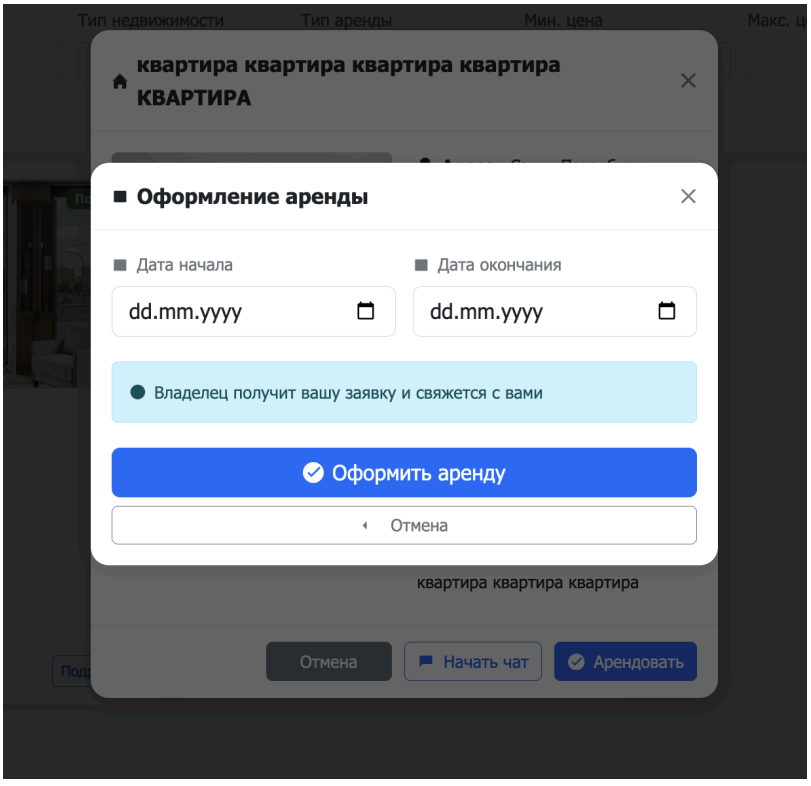

Рисунок 4 - Переход от деталей объекта недвижимости к созданию заявки на аренду

Вывод

Приложение было мигрированно с ванильной структуры на Vue, была реализован роутер, для работы с внешним API использована обертка с axios, элементы разделены на страницы, компоненты и composables.