САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Фронт-энд разработка

Отчет

Лабораторная работа №3

Выполнила:

Гусейнова Марьям

Группа ФРЭНД 2.2

Проверил:
Добряков Д. И.

Санкт-Петербург

2025 г.

Задача

Мигрировать ранее разработанное приложение (в рамках ЛР1 и ЛР2) на фреймворк Vue.JS.

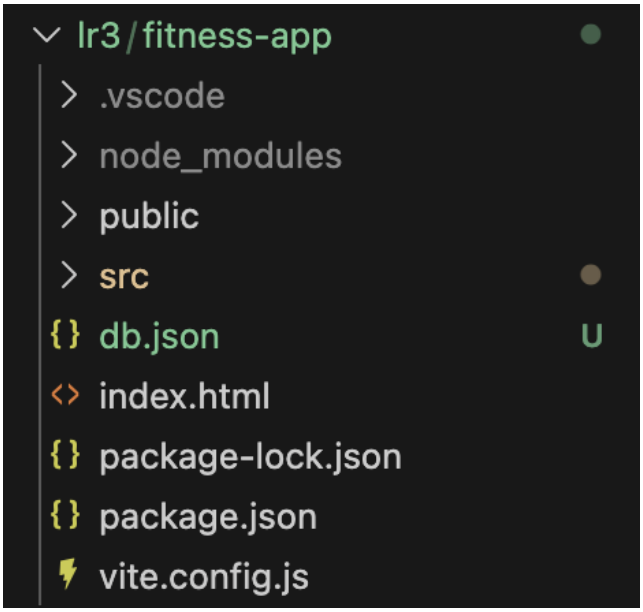
Каким требованиям приложение должно соответствовать:

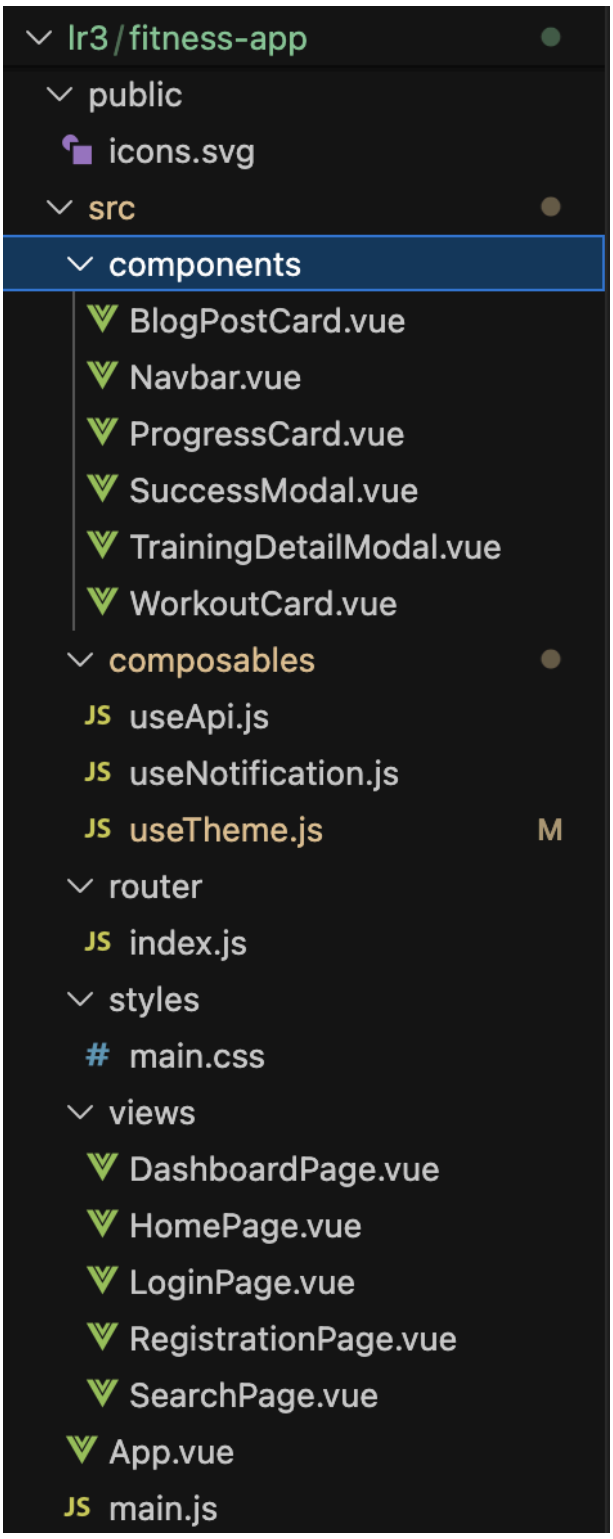
- Должен быть подключён роутер
- Должна быть реализована работа с внешним API (желательно посредством axios)
- Разумное деление на компоненты (продемонстрировать понимание компонентного подхода)
- Использование composable для выделения повторяющегося функционала в отдельные файлы.

Ход работы

В начале работы был инициализирован проект Vue 3 с использованием Vite, установлены необходимые зависимости: vue, vue-router, axios, bootstrap и настроена структура проекта согласно компонентному подходу.

Структура проекта:





Ниже описание каждого файла.

Корневые файлы:

package.json – файл конфигурации проекта, содержащий:

- список зависимостей: Vue 3.3.4, Vue Router 4.2.4, axios 1.5.0, Bootstrap 5.3.1;
- скрипты для запуска разработки (npm run dev), сборки (npm run build) и предпросмотра (npm run preview);
- настройки модульной системы (type: "module") для использования ES6 модулей.

vite.config.js – конфигурация сборщика Vite:

- подключение плагина Vue для обработки .vue файлов;
- настройка порта разработки (5173);
- оптимизация сборки для production и development режимов.

db.json – база данных для JSON-server:

- содержит три коллекции: users (пользователи), workouts (тренировки), blogPosts (статьи блога);
- каждая запись имеет уникальный ID и структурированные данные;
- используется как mock API сервер на localhost:3000;
- поддерживает CRUD операции и фильтрацию через query-параметры.

index.html – основной HTML файл:

- содержит базовую разметку страницы;
- подключает Bootstrap CSS для стилизации;
- имеет корневой элемент для монтирования Vue приложения;
- загружает главный JavaScript файл как ES6 модуль.

public/icons.svg – SVG-спрайт с иконками:

- содержит все векторные иконки, используемые в приложении;
- каждая иконка определена как <symbol> с уникальным ID;
- используется через <use> элементы для оптимизации загрузки;
- включает иконки: логотип, поиск, пользователь, вход, луна/солнце (переключение тем), график, календарь, галочка, крестик, гамбургер-меню.

Директория src/ – исходный код приложения:

Точка входа и настройки:

src/main.js – главный файл инициализации:

- создает экземпляр Vue приложения с помощью createApp();
- подключает Vue Router для управления маршрутизацией;
- импортирует стили Bootstrap и глобальные CSS стили;
- монтирует приложение в DOM элемент #app;
- служит центральной точкой конфигурации всех глобальных зависимостей.

src/App.vue – корневой компонент приложения:

- содержит общую структуру приложения: навигацию и основной контент;
- использует <router-view> для отображения текущей страницы;
- предоставляет модальные окна (SuccessModal и TrainingDetailModal) через provide/inject API;
- управляет состоянием модальных окон на уровне всего приложения;
- служит оберткой для всех страниц и компонентов.

Маршрутизация (src/router/):

src/router/index.js – конфигурация маршрутизатора:

- определяет все маршруты приложения с использованием createRouter() и createWebHistory();
- каждый маршрут связывает URL путь с соответствующим Vue компонентом-страницей;
- реализует навигацию между: главной страницей (/), поиском (/search), личным кабинетом (/dashboard), входом (/login), регистрацией (/register);
- использует HTML5 History API для чистых URL без хэшей;
- экспортирует настроенный экземпляр роутера для использования в основном приложении.

Composables (src/composables/) – переиспользуемая логика:

src/composables/useApi.js – управление HTTP-запросами:

- инкапсулирует всю логику работы с API через библиотеку axios;

- предоставляет реактивные состояния loading и error для отслеживания состояния запросов;
- реализует методы get() и post() для выполнения соответствующих HTTP-запросов;
- обрабатывает ошибки сети и сервера, пробрасывая их в компоненты;
- использует базовый URL http://localhost:3000 для JSON-server;
- переиспользуется во всех компонентах, требующих загрузки данных.

src/composables/useNotification.js – управление уведомлениями:

- предоставляет единый интерфейс для показа успешных сообщений и ошибок;
- использует механизм provide/inject для доступа к модальным окнам;
- имеет методы showSuccess() для показа успешных операций и showError() для ошибок;
- поддерживает автоматическую навигацию после закрытия уведомления;
- централизует логику отображения пользовательских сообщений.

src/composables/useTheme.js – управление цветовой темой:

- реализует логику переключения между светлой и темной темами;
- проверяет сохраненные настройки пользователя в localStorage;
- определяет системную тему через window.matchMedia('(prefers-color-scheme: dark)');
- обновляет классы CSS на элементе body при изменении темы;
- предоставляет реактивное состояние isDark и метод toggleTheme();
- сохраняет выбор пользователя только при явном переключении темы.

Компоненты (src/components/) – переиспользуемые UI элементы:

src/components/Navbar.vue – навигационная панель:

- содержит логотип, меню навигации и кнопку переключения темы;
- использует <router-link> для навигации между страницами;
- применяет динамические классы для активных ссылок и кнопки темы;
- интегрируется с useTheme() composable для управления темой;
- адаптивный дизайн с гамбургер-меню для мобильных устройств;

- имеет aria-атрибуты для доступности навигации.

src/components/SuccessModal.vue – модальное окно уведомлений:

- универсальный компонент для показа успешных операций и ошибок;
- использует Bootstrap Modal для отображения;
- динамически меняет цвета и иконки в зависимости от типа сообщения;
- поддерживает автоматическую навигацию после закрытия;
- интегрируется с Bootstrap через JavaScript API;
- имеет состояния успех (зеленый) и ошибка (красный).

src/components/TrainingDetailModal.vue – модальное окно деталей тренировки:

- показывает подробную информацию о выбранной тренировке;
- включает видео-инструкцию через iframe с YouTube;
- отображает пошаговые инструкции выполнения упражнений;
- имеет кнопку "Начать тренировку" для перехода к выполнению;
- управляется через Bootstrap Modal API;
- поддерживает доступность с aria-атрибутами.

src/components/WorkoutCard.vue – карточка тренировки:

- отображает информацию о тренировке: название, описание, тип, уровень, длительность;
- генерирует динамическое изображение с цветом в зависимости от типа тренировки;
- имеет кнопку "Подробнее" для открытия деталей тренировки;
- используется на странице поиска для отображения результатов;
- принимает данные тренировки через props;
- излучает событие show-details при клике на кнопку.

src/components/BlogPostCard.vue – карточка статьи блога:

- отображает статьи о здоровье и питании на главной странице;
- показывает изображение, заголовок и краткое описание статьи;
- имеет кнопку "Читать" для перехода к полной статье;
- принимает данные статьи через props;
- соответствует требованиям доступности.

src/components/ProgressCard.vue – карточка прогресса:

- отображает статистику пользователя в личном кабинете;
- показывает вес, количество тренировок за неделю, прогресс по отжиманиям;
- использует бейджи для визуального отображения изменений;
- имеет кнопку "Обновить данные" для ручного обновления статистики;
- получает данные прогресса через props от родительского компонента;
- интегрируется с системой иконок через SVG спрайт.

Страницы (src/views/) – основные экраны приложения:

src/views/HomePage.vue – главная страница:

- содержит секцию со слоганом и кнопками навигации;
- загружает и отображает статьи блога через API запрос;
- использует BlogPostCard компоненты для отображения статей;
- выполняет загрузку данных при монтировании компонента;
- предоставляет навигационные ссылки;
- служит точкой входа в приложение.

src/views/LoginPage.vue – страница входа:

- содержит форму для авторизации с полями email и пароль;
- выполняет валидацию учетных данных через API запрос;
- сохраняет данные пользователя в localStorage при успешной авторизации;
- использует useNotification для показа сообщений об успехе/ошибке;
- перенаправляет пользователя в личный кабинет после входа;
- имеет ссылку на страницу регистрации для новых пользователей.

src/views/RegistrationPage.vue – страница регистрации:

- содержит форму регистрации с полями: имя, email, пароль, подтверждение пароля;
- выполняет валидацию паролей на совпадение;
- проверяет уникальность email через API запрос;
- создает нового пользователя в системе через POST запрос;

- использует useNotification для обратной связи с пользователем;
- перенаправляет на страницу входа после успешной регистрации.

src/views/DashboardPage.vue – личный кабинет:

- отображает персональные данные и статистику пользователя;
- загружает информацию пользователя через API по ID из localStorage;
- использует ProgressCard для отображения статистики;
- показывает план тренировок в виде Bootstrap аккордеона;
- обновляет данные при каждом посещении страницы;
- предоставляет функционал для изменения плана тренировок.

src/views/SearchPage.vue – страница поиска тренировок:

- реализует систему фильтрации тренировок с параметрами: уровень, тип, максимальная длительность;
- выполняет серверную фильтрацию через query-параметры API запроса;
- отображает результаты поиска с использованием WorkoutCard компонентов;
- интегрируется с TrainingDetailModal для показа деталей тренировки;
- показывает количество найденных результатов;
- поддерживает индикатор загрузки во время выполнения запросов.

Стили (src/styles/):

src/styles/main.css – глобальные стили приложения:

- определяет CSS переменные для светлой и темной тем;
- содержит стили для всех компонентов и страниц;
- реализует плавные переходы при изменении темы;
- обеспечивает адаптивный дизайн для мобильных устройств;
- содержит специфичные стили для темной темы;
- определяет кастомные стили для брендинга приложения;
- поддерживает доступность через соответствующие CSS правила.

Подводя **итоги**:

Приложение разделено на 12 логически обособленных компонентов. Каждый компонент имеет четкую ответственность и переиспользуемость.

Используется однонаправленный поток данных через props и события. Компоненты изолированы и могут быть протестированы независимо.

Вся бизнес-логика вынесена в отдельные composable функции. Состояние управляется реактивно через ref() и computed(). Логика загрузки данных, уведомлений и темы изолирована в отдельных модулях.

Приложение соответствует правилам доступности. Все интерактивные элементы имеют aria-атрибуты. Используется семантическая разметка HTML5. Реализована клавиатурная навигация по модальным окнам. Цветовые контрасты соответствуют WCAG стандартам. Имеются альтернативные тексты для всех изображений.

Системная тема определяется автоматически при первом посещении. Выбор пользователя сохраняется в localStorage. Применяются CSS переменные для динамического изменения цветов. Все компоненты реагируют на изменение темы через CSS классы.

Вывод

В ходе выполнения лабораторной работы было успешно реализовано одностраничное веб-приложение для управления фитнес-тренировками с использованием фреймворка Vue.js 3. Приложение демонстрирует:

- полноценную архитектуру SPA с клиентской маршрутизацией через Vue Router;
- компонентный подход с четким разделением ответственности между компонентами;
- использование Composition API для выделения повторяющейся логики в composable функции;
- интеграцию с внешним API через axios с серверной фильтрацией данных;
- соблюдение принципов доступности и поддержку темной/светлой темы.