React - TODO

проектная работа

Тех задание к проекту (приложение TODO)

Стек:

- Webpack/Vite
- JavaScript
- React 18
- React-router-dom 6 (библиотека для маршрутизации)
- Redux (библиотека для управления состоянием)
- Redux thunk (асинхронный middleware для Redux)
- Axios (библиотека HTTP запросов)
- React-bootstrap (UI библиотека)

Реализовать SPA приложение для работы с TODO с помощью React, получение данных с сервера выполнить с помощью axios, синхронизировать состояние списка TODO полученного с сервера с помощью Redux, последующие взаимодействия с сервером синхронизировать с приложением через Redux Thunk. Визуал (UI) свободного выбора, но реализовать с помощью UI библиотеки React Bootstrap. Маршрутизацию в приложении реализовать с помощью React Router DOM

Сущность TODO и ее функционал

Отображение списка

Добавление новой записи в список

Удаление существующей записи

Изменение статуса записи (поле “completed”)

Поиск по имени (поле “title”) необходимого todo из списка todo

При создании новой записи, id должен генерироваться, по дефолту completed у новых задач всегда false, userId можно пренебречь (случайное значение)

Todo: {
  "userId": <number>,
  "id": <number>,
  "title": <string>,
  "completed": <boolean>
}

TodoList: Array<Todo>

userId - пренебрегаем

API

Основной URL мокового back-end-а: ***https://jsonplaceholder.typicode.com***

Методы:

- GET ***/todos*** — получение списка всех задач.
- GET ***/todos/:id*** — получение задачи по ID.
- POST ***/todos*** — добавление новой задачи.
- PUT ***/todos/:id*** — обновление задачи по ID.
- DELETE ***/todos/:id*** — удаление задачи по ID.

Особенность:
Одним из ключевых недочетов **JSONPlaceholder** является то, что он предоставляет фиктивные данные, и внесенные изменения, такие как добавление, обновление или удаление задач, не сохраняются на сервере. Все запросы на изменение данных (POST, PUT, DELETE) возвращают успешный ответ, но фактически данные не изменяются на сервере.

UI

Для визуального оформления использовать UI библиотеку Bootstrap

Ключевые моменты:

Шапка должна оставаться на виду у пользователя при скроле

В шапке должна присутствовать навигация

Текущий маршрут (путь/url) должен выделяться в навигации

В навигации должна присутствовать кнопка смены темы

Карточка TODO должна визуально отображать статус задачи, ее название, а также предоставлять возможность удаления и смены статуса через кнопки в карточке

Список должен состоять из карточек, список должен поддерживать асинхронность, то есть должны быть состояния: пустой список, загрузка, ошибка, отображение списка.

Для добавления TODO должно присутствовать модальное окно с формой, input для названия

Страницы

Главная (main page) - Главная страница кнопка создания нового TODO и отображение списка всех задач

Выполненные (completed todo page) - отображение всех выполненных задач

Невыполненные (uncompleted todo page) - отображение всех невыполненных задач

О проекте (about page) - страница с описанием проекта

404 (error page) - страница для отловки ошибки маршрутизации

Дополнительно к заданию

В readme файле описать свой проект

Выполненный проект разделить на 2 части

Исходный код - залить на git репозиторий

Результат - деплой на стороннем сервиса githubPages/vercel/netlify

Критерии оценки

Чистота кода

Правильное использование технологий

Красивый визуал (UI)

Адаптивность

Грамотная архитектура

Корректные именования

Ресурсы

react bootstrap - ТЫК

документация React- ТЫК

документация React Router DOM - ТЫК

документация Axios - ТЫК

документация Redux - ТЫК

документация Redux thunk - ТЫК

vercel - ТЫК