PostPageBeforeVueX.vue

<template>

    <div>

        <MyInput

            v-model="searchQuery"

            placeholder="Поиск"

        />

        <div class="app\_btns">

            <MyButton

                @click="showDialog"

            >

                Создать пост

            </MyButton>

            <MySelect

                v-model="selectedSort"

                :options="sortOptions"

            />

        </div>

        <MyDialog v-model:show="dialogVisible">

            <PostForm

                @create="createPost"

            />

        </MyDialog>

        <PostList

            :posts="sortedAndSearhedPosts"

            @remove="removePost"

            v-if="!isPostLoading"

        />

        <div v-else>Загрузка...</div>

        <div ref="observer"></div>

    </div>

    <!-- прослушиваем событи от ребенка @remove и вызываем на него соответствующую ф-ю-->

</template>

<script>

import PostForm from '@/components/PostForm.vue';

import PostList from '@/components/PostList.vue';

import MyDialog from '@/components/UI/MyDialog.vue';

import MyButton from '@/components/UI/MyButton.vue';

import MySelect from "@/components/UI/MySelect";

import MyInput from "@/components/UI/MyInput";

import axios from 'axios';

export default {

    name: 'PostPageBeforeVueX',

    components: { PostForm, PostList, MyDialog, MyButton, MySelect, MyInput },

    data() {

        return {

            posts: [

            // {id: 1, title: 'заголовок 1', body: 'текст поста 1'},

            // {id: 2, title: 'заголовок 2', body: 'текст поста 2'},

            // {id: 3, title: 'заголовок 3', body: 'текст поста 3'},

            ],

            dialogVisible: false,

            isPostLoading: false, // индикатор загрузки - может заменить на крутилку?

            selectedSort: '',

            searchQuery: '',

            page: 1,

            limit: 10,

            totalPages: 0,

            sortOptions: [

                {value: 'title', name: 'По названию'},

                {value: 'body', name: 'По содержанию'},

            // {value: 'id', name: 'По дате'},   // добавить сортировку по дате?

            ]

        }

    },

    methods: {

        createPost(post) {

            this.posts.push(post);

            this.dialogVisible = false;

        },

        removePost(post) {

            this.posts = this.posts.filter(p => p.id !== post.id)

        },

        showDialog() {

            this.dialogVisible = true;

        },

        // changePage(pageNumber) {

        //   this.page = pageNumber;

        //   this.fetchPosts(); // можно сделать в watch

        // },

        // https://ru.vuejs.org/v2/cookbook/using-axios-to-consume-apis.html

        async fetchPosts() {

            try {

                this.isPostLoading = true;

                // this.$refs.observer                       //////////////////////// ??????????????????

            // делаем запрос на сервис

            const response = await axios.get('https://jsonplaceholder.typicode.com/posts', {

                params: {

                    \_page: this.page,

                    \_limit: this.limit

                }

            });

            // высчитываем ск-ко страниц. Общее кол-во постов делим на кол-во постов на 1 странице, округляя в большую сторону (101 / 10 = 11)

                this.totalPages = Math.ceil(response.headers['x-total-count'] / this.limit)

                this.posts = response.data;

            } catch (error) {

                alert('Ошибка');

            } finally {

                this.isPostLoading = false;

            }

        },

        async loadMorePosts() {

            try {

            this.page += 1;

            // this.isPostLoading = true;

            // делаем запрос на сервис

            const response = await axios.get('https://jsonplaceholder.typicode.com/posts', {

                params: {

                    \_page: this.page,

                    \_limit: this.limit

                }

            });

            // высчитываем ск-ко страниц. Общее кол-во постов делим на кол-во постов на 1 странице, округляя в большую сторону (101 / 10 = 11)

            this.totalPages = Math.ceil(response.headers['x-total-count'] / this.limit)

                this.posts = [...this.posts, ...response.data];

            } catch (error) {

                alert('Ошибка');

            }

        // finally {

        //   this.isPostLoading = false;

        // }

        },

    },

    mounted() {

        this.fetchPosts();

        // https://developer.mozilla.org/en-US/docs/Web/API/Intersection\_Observer\_API

        const options = {

            // root: document.querySelector("#scrollArea"),

            rootMargin: "0px",

            threshold: 1.0,

        };

        const callback = (entries, observer) => {

            if(entries[0].isIntersecting && this.page < this.totalPages) {

                this.loadMorePosts();

            }

        };

        const observer = new IntersectionObserver(callback, options);

        observer.observe(this.$refs.observer);

    },

    computed: {

        // используем эту ф-ю как переменную, земенив :posts="posts" в PostList на :posts="sortedPosts"

        sortedPosts() {

            return [...this.posts].sort((post1, post2) => post1[this.selectedSort]?.localeCompare(post2[this.selectedSort]))

        },

        sortedAndSearhedPosts() {

            return this.sortedPosts.filter(post => post.title.toLowerCase().includes(this.searchQuery.toLowerCase()))

        }

    },

    watch: {

      // ф-я наблюдатель отрабатывает на смену страницы

      // page() {

      //   this.fetchPosts()

      // }

      //-----------------

      // selectedSort(newValue) {

      //   this.posts.sort((post1, post2) => {

      //     // return post1[this.selectedSort] ?. localCompare(post2[this.selectedSort]);

      //     return post1[newValue] ?.localeCompare(post2[newValue])

      //   });

      // }

    }

}

</script>

<style lang="scss">

  \* {

    margin: 0;

    padding: 0;

    box-sizing: border-box;

}

.app {

    padding: 12px;

    &\_btns {

        display: flex;

        justify-content: space-between;

        margin: 15px, 0;

    }

}

</style>