UNIVERSITATEA DE STAT DIN MOLDOVA
Specialitatea: «Informatica aplicată»
Disciplina: «React»

Лабораторная работа №2

A verificat:
Denis Negura

A elaborat:
Ianciuc Nadejda

Chișinău, 2025

План

План **2**
Цели работы **3**
Реализация требований **3**
1. Поиск по названию и описанию 3
2. Панели фильтров 4
3. Оптимизация вычислений useMemo 8
Результаты (скриншоты) **12**
Вывод **13**

Цели работы

1. Научиться создавать поиск по данным (по названию/полям).
2. Реализовать фильтры (сортировка, диапазоны, чекбоксы и т.п.)
3. Освоить оптимизацию рендера: мемоизация результатов фильтрации.

Реализация требований

1. Поиск по названию и описанию

Компонент поиска реализован как контролируемый элемент, состояние которого полностью управляется родительским компонентом. Это обеспечивает синхронное обновление UI и логики фильтрации без задержек. Очистка поля реализована кнопкой, что повышает удобство пользователя.

Файл: src/components/SearchBar.jsx

```
import React from 'react';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import * as Icons from '@fortawesome/free-solid-svg-icons';

const SearchBar = ({ searchQuery, onSearchChange }) => {
    return (
        <div className="relative flex-1">
            <div className="relative">
                <FontAwesomeIcon
                    icon={Icons.faSearch}
                            className="absolute left-4 top-1/2 transform
-translate-y-1/2"
                />
                <input
                    type="text"
                    value={searchQuery}
                    onChange={(e) => onSearchChange(e.target.value)}
                    placeholder="Поиск покемонов..."
                           className="w-full pl-12 pr-10 py-3 rounded-lg
border-2
                               bg-white/90 focus:border-yellow-400"
                />
```

```
                {searchQuery && (
                    <button
                        onClick={() => onSearchChange('')}
                            className="absolute right-3 top-1/2 transform
-translate-y-1/2"
                    >
                        <FontAwesomeIcon icon={Icons.faTimes} />
                    </button>
                )}
            </div>
        </div>
    );
};
```

2. Панели фильтров

Панель фильтрации включает:

- сортировку
- выбор типов (множественный селект через чекбоксы/кнопки)
- фильтрацию по диапазонам
- счётчик найденных элементов

Типы визуально выделены цветами, что улучшает UX. Диапазоны реализованы с помощью input-range.

Файл: src/components/FilterPanel.jsx

```
const FilterPanel = ({
    filters,
    onFilterChange,
    onResetFilters,
    availableTypes,
    foundCount
}) => {
    const pokemonTypes = availableTypes || [
        'normal', 'fire', 'water', 'electric', 'grass', 'ice',
        'fighting', 'poison', 'ground', 'flying', 'psychic',
        'bug', 'rock', 'ghost', 'dragon', 'dark', 'steel', 'fairy'
    ];
```

```
    const sortOptions = [
        { value: 'name-asc', label: 'Имя (A-Z) (А-Я)' },
        { value: 'name-desc', label: 'Имя (Z-A) (Я-А)' },
        { value: 'height-asc', label: 'Рост (возр.)' },
        { value: 'height-desc', label: 'Рост (убыв.)' },
        { value: 'weight-asc', label: 'Вес (возр.)' },
        { value: 'weight-desc', label: 'Вес (убыв.)' },
    ];

    const handleTypeToggle = (type) => {
        const newTypes = filters.types.includes(type)
            ? filters.types.filter(t => t !== type)
            : [...filters.types, type];
        onFilterChange({ ...filters, types: newTypes });
    };

    const getTypeColor = (type) => {
        const colors = {
            normal: 'bg-gray-400',
            fire: 'bg-red-500',
            water: 'bg-blue-500',
            electric: 'bg-yellow-400',
            grass: 'bg-green-500',
            // ... остальные типы
        };
        return colors[type] || 'bg-gray-400';
    };

    return (
        <div className="bg-white/90 rounded-xl shadow-lg p-6">
            {/* Заголовок с счётчиком и кнопкой сброса */}
            <div className="flex justify-between items-center mb-6">
                    <div className="bg-green-100 text-green-800 px-4 py-2
rounded-lg">
                    <FontAwesomeIcon icon={Icons.faCheckCircle} />
                    Найдено: {foundCount}
                </div>
                <button
                    onClick={onResetFilters}
```

```
                        className="bg-red-500 hover:bg-red-600 text-white
px-4 py-2 rounded-lg"
                >
                    <FontAwesomeIcon icon={Icons.faUndo} />
                    Сбросить
                </button>
            </div>

            {/* 1. Сортировка */}
            <div className="mb-6">
                <label className="block text-sm font-semibold mb-2">
                    <FontAwesomeIcon icon={Icons.faSort} />
                    Сортировка
                </label>
                <select
                    value={filters.sortBy}
                    onChange={(e) => onFilterChange({
                        ...filters,
                        sortBy: e.target.value
                    })}
                    className="w-full px-4 py-3 rounded-lg border-2"
                >
                    {sortOptions.map(option => (
                        <option key={option.value} value={option.value}>
                            {option.label}
                        </option>
                    ))}
                </select>
            </div>

            {/* 2. Фильтр по типам (чекбоксы) */}
            <div className="mb-6">
                <label className="block text-sm font-semibold mb-3">
                    <FontAwesomeIcon icon={Icons.faTags} />
                    Типы покемонов ({filters.types.length} выбрано)
                </label>
                <div className="flex flex-wrap gap-2">
                    {pokemonTypes.map(type => (
                        <button
                            key={type}
```

```
                        onClick={() => handleTypeToggle(type)}
                        className={`${getTypeColor(type)} ${
                            filters.types.includes(type)
                                ? 'ring-4 ring-yellow-400 scale-105'
                                : 'opacity-50 hover:opacity-100'
                                 } text-white px-4 py-2 rounded-full
uppercase`}
                    >
                        {filters.types.includes(type) && (
                            <FontAwesomeIcon icon={Icons.faCheck} />
                        )}
                        {type}
                    </button>
                ))}
            </div>
        </div>

        {/* 3. Диапазон роста (ползунок) */}
        <div className="mb-6">
            <label className="block text-sm font-semibold mb-2">
                Максимальный рост: {filters.heightRange[1]} дм
                ({filters.heightRange[1] / 10}м)
            </label>
            <input
                type="range"
                min="0"
                max="100"
                value={filters.heightRange[1]}
                onChange={(e) => onFilterChange({
                    ...filters,
                    heightRange: [0, parseInt(e.target.value)]
                })}
                className="w-full"
            />
        </div>

        {/* 4. Диапазон веса (ползунок) */}
        <div className="mb-6">
            <label className="block text-sm font-semibold mb-2">
                Максимальный вес: {filters.weightRange[1]} гг
```

```
                    ({filters.weightRange[1] / 10}кг)
                </label>
                <input
                    type="range"
                    min="0"
                    max="1000"
                    value={filters.weightRange[1]}
                    onChange={(e) => onFilterChange({
                        ...filters,
                        weightRange: [0, parseInt(e.target.value)]
                    })}
                    className="w-full"
                />
            </div>
        </div>
    );
};

export default FilterPanel;
```

3. Оптимизация вычислений useMemo

Мемоизация используется для:

- фильтрации (зависимости: данные, фильтры, поисковый запрос)
- пагинации (зависимости: отфильтрованные данные и текущая страница)

Это позволяет выполнять пересчёт только при необходимости, снижая количество операций сортировки и фильтрации.

Файл: src/pages/HomePage.jsx

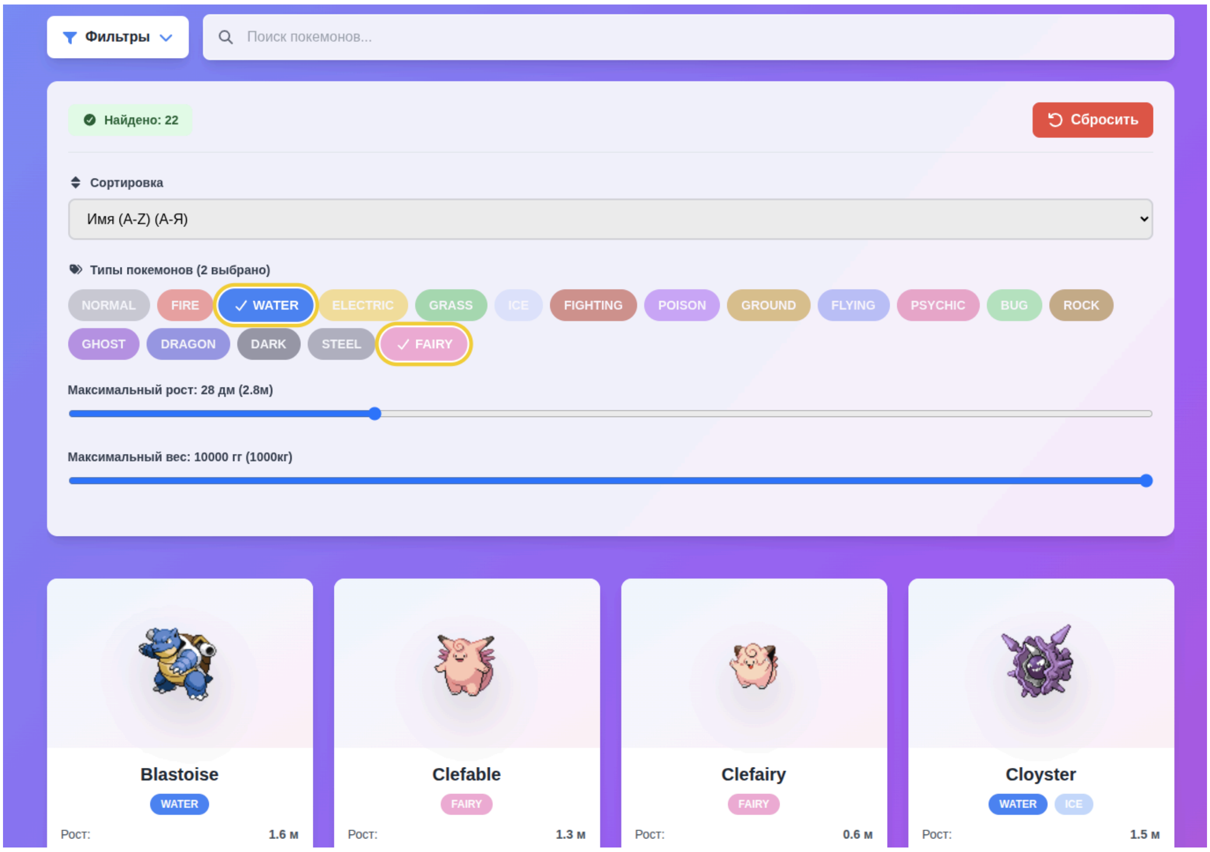
```
const HomePage = () => {
    const { pokemons, loading, ITEMS_PER_PAGE } = usePokemon();

    // Состояния
    const [searchQuery, setSearchQuery] = useState('');
    const [currentPage, setCurrentPage] = useState(1);
    const [filters, setFilters] = useState({
        types: [],
        sortBy: 'name-asc',
        heightRange: [0, 200],
```

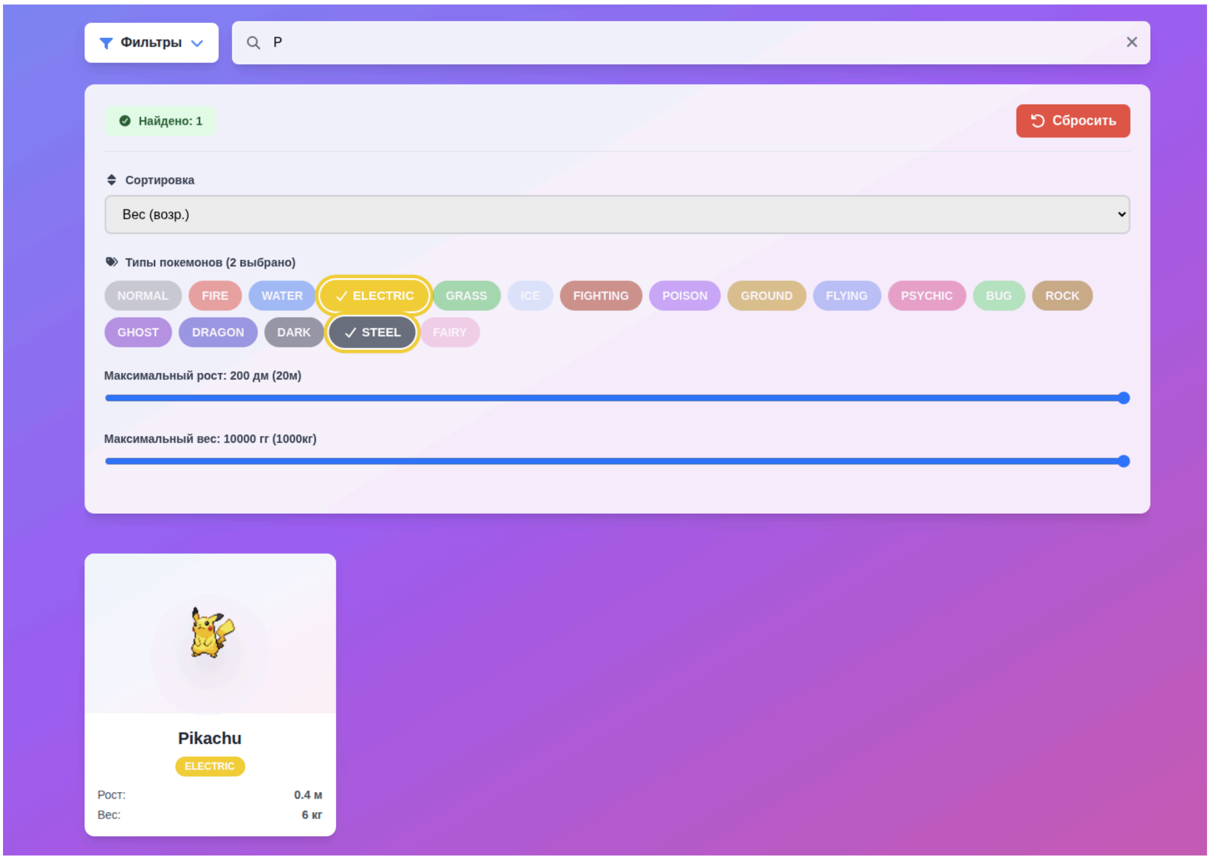
```
        weightRange: [0, 10000],
    });

    // МЕМОИЗИРОВАННАЯ ФИЛЬТРАЦИЯ
    const filteredPokemons = useMemo(() => {
        let result = [...pokemons];

        // 1. Поиск по имени
        if (searchQuery) {
            result = result.filter(pokemon =>

pokemon.name.toLowerCase().includes(searchQuery.toLowerCase())
            );
        }

        // 2. Фильтр по типам
        if (filters.types.length > 0) {
            result = result.filter(pokemon =>
                pokemon.types.some(type =>
                    filters.types.includes(type.type.name)
                )
            );
        }

        // 3. Фильтр по росту
        result = result.filter(pokemon =>
            pokemon.height >= filters.heightRange[0] &&
            pokemon.height <= filters.heightRange[1]
        );

        // 4. Фильтр по весу
        result = result.filter(pokemon =>
            pokemon.weight >= filters.weightRange[0] &&
            pokemon.weight <= filters.weightRange[1]
        );

        // 5. Сортировка
        const [sortField, sortOrder] = filters.sortBy.split('-');
        result.sort((a, b) => {
            let aValue = a[sortField];
```

```
            let bValue = b[sortField];

            if (sortField === 'name') {
                aValue = a.name.toLowerCase();
                bValue = b.name.toLowerCase();
                return sortOrder === 'asc'
                    ? aValue.localeCompare(bValue)
                    : bValue.localeCompare(aValue);
            }

                return sortOrder === 'asc' ? aValue - bValue : bValue -
aValue;
        });

        return result;
    }, [pokemons, searchQuery, filters]);

    // МЕМОИЗИРОВАННАЯ ПАГИНАЦИЯ
    const paginatedPokemons = useMemo(() => {
        const startIndex = (currentPage - 1) * ITEMS_PER_PAGE;
        const endIndex = startIndex + ITEMS_PER_PAGE;
        return filteredPokemons.slice(startIndex, endIndex);
    }, [filteredPokemons, currentPage, ITEMS_PER_PAGE]);

    const calculatedTotalPages = Math.ceil(
        filteredPokemons.length / ITEMS_PER_PAGE
    ) || 1;

    const handleResetFilters = () => {
        setFilters({
            types: [],
            sortBy: 'name-asc',
            heightRange: [0, 200],
            weightRange: [0, 10000],
        });
        setCurrentPage(1);
    };

    return (
        <div>
```

```
            <SearchBar
                searchQuery={searchQuery}
                onSearchChange={setSearchQuery}
            />

            <FilterPanel
                filters={filters}
                onFilterChange={setFilters}
                onResetFilters={handleResetFilters}
                foundCount={filteredPokemons.length}
            />

            <PokemonList
                pokemons={paginatedPokemons}
                loading={loading}
                currentPage={currentPage}
                totalPages={calculatedTotalPages}
                onPageChange={setCurrentPage}
            />
        </div>
    );
};
```

Результаты (скриншоты)





Вывод

В результате выполнения лабораторной работы были достигнуты следующие результаты:

- реализован поиск по данным;
- добавлена панель фильтров с несколькими типами критериев;
- выполнена оптимизация через useMemo;
- реализована пагинация;
- созданы визуальные компоненты для отображения состояний;
- обеспечено мгновенное обновление интерфейса при изменении фильтров.

В ходе лабораторной работы была освоена практика работы с фильтрами, поиском и клиентской оптимизацией данных. Реализация на React позволила применить современные подходы к управлению состоянием и рендером. Система поиска и фильтрации получилась гибкой, расширяемой и производительной, что соответствует требованиям к современным веб-приложениям.