МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ

образовательное учреждение

высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра автоматизированных систем управления



**ОТЧЁТ**

**по лабораторной работе №13**

**на тему:**

**Основы React**

**по дисциплине:** Разработка программных приложений и WEB-программирование

Выполнили работу:

Студенты гр. АП-226, АВТФ

Ищенко С. Г.

«07» июня 2025 г.

Проверил работу:

Эстрайх И. В.

«\_\_\_» \_\_\_\_\_\_ 20\_\_г

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

**Цель**

Получить практические навыки разработки веб-приложений с

помощью библиотеки React.

**Задание**

1. Установить все необходимые инструменты для разработки

приложений на React.

2. Создать на основе компонентов React веб-страницу аналогичную

разработанной в течение лабораторных работ 5-12, но без бэкенда

(отсутствует авторизация, БД и т.д.).

3. Обеспечить сортировку записей на странице. Использовать ключи

для предотвращения ненужной перерисовки компонентов.

4. Добавить на страницу форму для добавления записей.

Предусмотреть валидацию данных формы с визуальным

оповещением пользователя о некорректности данных путем смены

стиля (например, другой цвет границы компонента, фона и др.).

**Описание хода выполнения работы**

1. Установить все необходимые инструменты для разработки

приложений на React.

Для начала в терминале нужно ввести команду установки в терминал

winget search nodejs

winget install OpenJS.NodeJS.LTS

или скачать с сайта https://nodejs.org/en/download/

После установки проверить установку и версии с помощью команд

node -v

npm -v

2. Создать на основе компонентов React веб-страницу аналогичную

разработанной в течение лабораторных работ 5-12, но без бэкенда

(отсутствует авторизация, БД и т.д.).

Всё, что нам нужно для «чистого» React-проекта, уже есть внутри Node.js/npm. Дальше мы воспользуемся инструментом create-react-app, который автоматически настроит webpack, Babel, ESLint и все прочие зависимости.

Для этого введем команду в терминал:

npx create-react-app .

Пояснение:

* + npx автоматически подгрузит последнюю версию пакета create-react-app (не нужно ничего устанавливать глобально).
  + Точка . в конце говорит «сгенерировать проект прямо в текущей папке», а не создавать ещё одну вложенную папку.

В течение минуты npm скачает все библиотеки и создаст стандартную структуру.

Сразу после этого запустим сервер c помощью команды: npm start , чтобы убедиться в корректности.

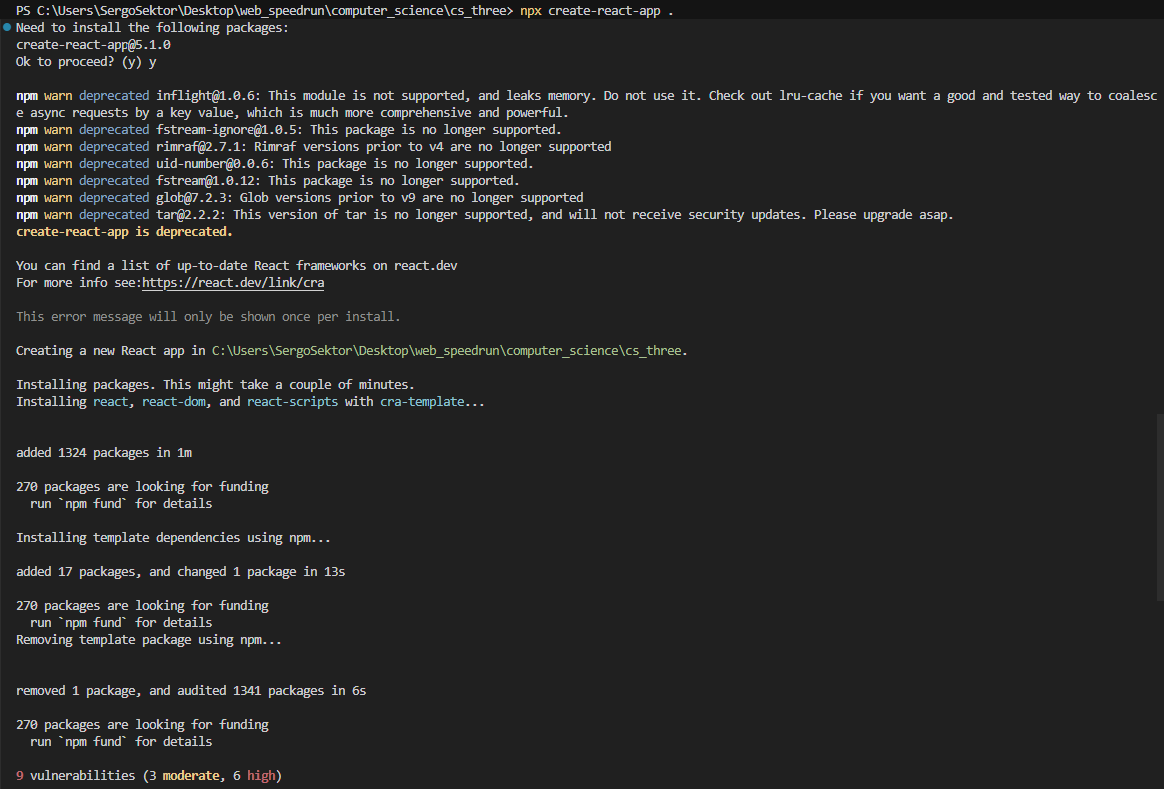
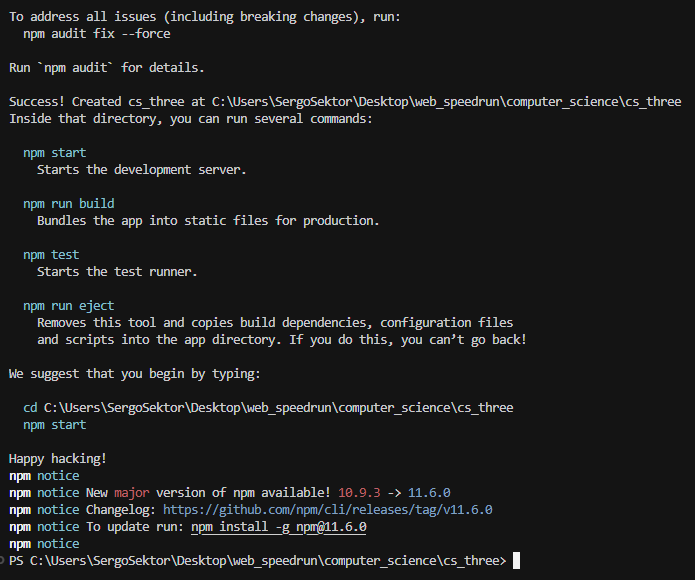


Рисунок 1 – установка npx

Рисунок 2 - Установка npx.

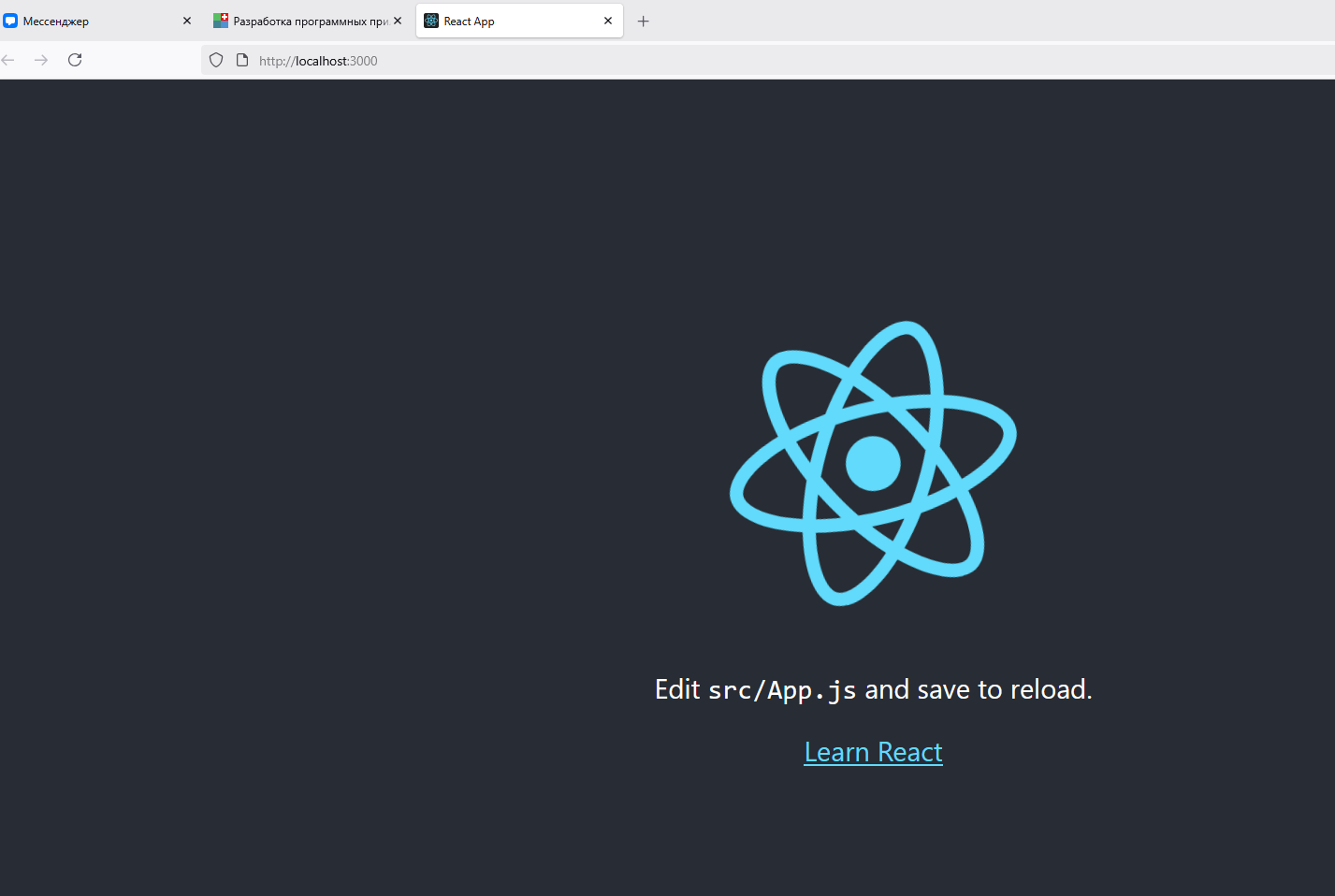


Рисунок 3 – Запуск сервера.

Нам нужно создать новую подпапку для наших компонентов. Щёлкним правой кнопкой мыши на папке src/ и выберем New → Directory (Новая папка). Назовем её components.

Внутри components/ создаем четыре файла:

* + ConceptItem.js
  + ConceptList.js
  + ConceptForm.js
  + styles.css

Заполним ConceptItem.js

import React from 'react';

function **ConceptItem**({ *concept* }) {

  return (

    <li *className*="concept-item">

      <h3>{*concept*.name}</h3>

      <p>{*concept*.definition}</p>

      <p>Уровень сложности: {*concept*.level}</p>

    </li>

  );

}

export default ConceptItem;

* Для чего:
  + Мы выносим визуализацию одной машины в отдельный компонент, чтобы весь список не превращался в один монолитный JSX-блок.
  + В ConceptList потом будем писать <ConceptItem key={concept.id} concept ={ concept } />, и React поймёт, что если id не меняется, то этот элемент можно не перерисовывать.

Заполним ConceptList.js

import React, { useState, useMemo } from 'react';

import ConceptItem from './ConceptItem';

function **ConceptList**({ *concepts* }) {

  const [sortKey, **setSortKey**] = useState('name');

  const sortedConcepts = useMemo(() => {

    let sorted = [...*concepts*];

    if (sortKey === 'name') {

      sorted.sort((*a*, *b*) => *a*.name.localeCompare(*b*.name));

    }

    else if (sortKey === 'levelAsc') {

      sorted.sort((*a*, *b*) => *a*.level - *b*.level);

    } else if (sortKey === 'levelDesc') {

      sorted.sort((*a*, *b*) => *b*.level - *a*.level);

    }

    return sorted;

  }, [*concepts*, sortKey]);

  const [currentPage, **setCurrentPage**] = useState(1);

  const itemsPerPage = 3;

  const totalPages = Math.ceil(sortedConcepts.length / itemsPerPage);

  const **getPageRange** = () => {

    const delta = 1;

    const range = [];

    for (

      let num = Math.max(1, currentPage - delta);

      num <= Math.min(totalPages, currentPage + delta);

      num++

    ) {

      range.push(num);

    }

    return range;

  };

  const startIdx = (currentPage - 1) \* itemsPerPage;

  const endIdx = startIdx + itemsPerPage;

  const conceptsToDisplay = sortedConcepts.slice(startIdx, endIdx);

  return (

    <div>

      <div *className*="sort-container">

        <label>Сортировать по: </label>

        <select *value*={sortKey} *onChange*={(*e*) => setSortKey(*e*.target.value)}>

          <option *value*="name">Название (A→Z)</option>

          <option *value*="levelAsc">Уровень сложности (возр.→убыв.)</option>

          <option *value*="levelDesc">Уровень сложности (убыв.→возр.)</option>

        </select>

      </div>

      <ul *className*="list-articles">

        {conceptsToDisplay.map((*concept*) => (

          <ConceptItem *key*={*concept*.id} *concept*={*concept*} />

        ))}

      </ul>

      {totalPages > 1 && (

        <nav *className*="pagination-nav">

          {currentPage > 1 && (

            <button *onClick*={() => setCurrentPage((*prev*) => prev - 1)}>

              Prev

            </button>

          )}

          {getPageRange().map((*num*) =>

            num === currentPage ? (

              <span *key*={num} *className*="page-number active">

                {num}

              </span>

            ) : (

              <button *key*={num} *onClick*={() => setCurrentPage(num)}>

                {num}

              </button>

            )

          )}

          {currentPage < totalPages && (

            <button *onClick*={() => setCurrentPage((*prev*) => prev + 1)}>

              Next

            </button>

          )}

        </nav>

      )}

    </div>

  );

}

export default ConceptList;

* 1. useState(‘title’) — хранит текущий способ сортировки.
  2. useMemo(…) — вычисляет отсортированный массив только при изменении concepts или sortKey. Это экономит ресурсы, потому что без useMemo при каждом рендере происходил бы новый sort().
  3. key={ concept.id} — обязательное свойство, чтобы React мог эффективно обновлять список, зная «уникальный идентификатор» каждого элемента.
  4. Блок пагинации:
     + Если totalPages <= 1, ничего не рендерится (условие отсутствия навигации, как требует задание).
     + Текущий номер страницы выводится как <span> (без ссылки) — выполняется требование «отсутствие ссылки на текущий номер».
     + Блок отображает только номера в диапазоне [currentPage-1, currentPage, currentPage+1], ограниченные размером totalPages. Это соответствует требованию «небольшое число отображаемых номеров слева и справа».
     + Prev/Next показываются только если такие страницы существуют (проверка currentPage > 1 и currentPage < totalPages).

4. Добавить на страницу форму для добавления записей. Предусмотреть валидацию данных формы с визуальным оповещением пользователя о некорректности данных путем смены стиля (например, другой цвет границы компонента, фона и др.).

Заполним ConceptForm.js:

import React, { useState } from 'react';

function **ConceptForm**({ *onAddConcept* }) {

  const [name, **setName**] = useState('');

  const [definition, **setDefinition**] = useState('');

  const [level, **setLevel**] = useState('');

  const [errors, **setErrors**] = useState({});

  const **validate** = () => {

    const errs = {};

    if (!name.trim()) errs.name = 'Название не может быть пустым';

    if (level === '' || isNaN(level) || Number(level) < 0) {

      errs.level = 'Уровень сложности должен быть числом ≥ 0';

    }

    return errs;

  };

  const **handleSubmit** = (*e*) => {

*e*.preventDefault();

    const validationErrors = validate();

    if (Object.keys(validationErrors).length > 0) {

      setErrors(validationErrors);

      return;

    }

    const newConcept = {

      id: Date.now(),

      name: name.trim(),

      definition: definition.trim(),

      level: Number(level),

    };

    onAddConcept(newConcept);

    setName('');

    setDefinition('');

    setLevel('');

    setErrors({});

  };

  return (

    <form *className*="concept-form" *onSubmit*={handleSubmit}>

      <div *className*="form-group">

        <label>Название:</label>

        <input

*type*="text"

*value*={name}

*onChange*={(*e*) => setName(*e*.target.value)}

*className*={errors.name ? 'invalid' : ''}

        />

        {errors.name && <div *className*="error-text">{errors.name}</div>}

      </div>

      <div *className*="form-group">

        <label>Описание:</label>

        <textarea

*value*={definition}

*onChange*={(*e*) => setDefinition(e.target.value)}

        />

      </div>

      <div *className*="form-group">

        <label>Уровень сложности:</label>

        <input

*type*="number"

*value*={level}

*onChange*={(*e*) => setLevel(e.target.value)}

*className*={errors.level ? 'invalid' : ''}

        />

        {errors.level && <div *className*="error-text">{errors.level}</div>}

      </div>

      <button *type*="submit" *className*="btn-submit">

        Добавить концепцию

      </button>

    </form>

  );

}

export default ConceptForm;

* 1. Храним состояние трёх полей: title, description, price.
  2. При onSubmit вызывается validate(). Если найдены ошибки, setErrors(...) добавит их; поля, у которых есть ошибки, получат класс invalid, и под ними выведется сообщение (див с классом error-text).
  3. Если же валидация успех, формируется объект новой машины и передаётся вверх через onAddConcept(...) (связь пропов). После этого сбрасываются все поля и ошибки.

Заполним styles.css

body {

  font-family: 'Roboto', 'Helvetica Neue', Arial, sans-serif;

  line-height: 1.6;

  color: #333;

  background-color: #f8f8f8;

  margin: 0;

  padding: 20px;

}

.App {

  max-width: 800px;

  margin: 0 auto;

  background-color: #fff;

  padding: 30px;

  border-radius: 8px;

  box-shadow: 0 2px 10px rgba(0, 0, 0, 0.05);

}

h1 {

  color: #003366;

  font-size: 2.2em;

  margin-bottom: 30px;

  text-align: center;

}

.concept-form {

  width: 100%;

  margin-bottom: 30px;

  padding: 25px;

  border: 1px solid #e0e0e0;

  border-radius: 6px;

  background-color: #fcfcfc;

}

.form-group {

  margin-bottom: 20px;

}

.form-group label {

  display: block;

  font-weight: 600;

  margin-bottom: 8px;

  color: #555;

}

.form-group input,

.form-group textarea {

  width: calc(100% - 20px);

  padding: 10px;

  border: 1px solid #d0d0d0;

  border-radius: 4px;

  font-size: 1em;

}

.form-group textarea {

  min-height: 80px;

  resize: vertical;

}

.form-group input.invalid,

.form-group textarea.invalid {

  border-color: #e74c3c; */\* Красный для ошибок \*/*

  background-color: #fffafa;

}

.error-text {

  color: #e74c3c;

  font-size: 0.85em;

  margin-top: 5px;

}

.btn-submit {

  background-color: #0056b3; */\* Темно-синий \*/*

  color: #fff;

  border: none;

  padding: 10px 20px;

  border-radius: 5px;

  cursor: pointer;

  font-size: 1.1em;

  transition: background-color 0.2s ease;

}

.btn-submit:hover {

  background-color: #004085;

}

.list-articles {

  list-style: none;

  padding: 0;

  margin: 0;

}

.concept-item {

  border: 1px solid #e0e0e0;

  padding: 15px 20px;

  border-radius: 6px;

  margin-bottom: 15px;

  background-color: #fff;

  box-shadow: 0 1px 4px rgba(0, 0, 0, 0.03);

}

.concept-item h3 {

  color: #003366;

  margin-top: 0;

  margin-bottom: 8px;

  font-size: 1.5em;

}

.concept-item p {

  color: #666;

  margin-bottom: 5px;

}

.sort-container {

  display: flex;

  align-items: center;

  justify-content: flex-end; */\* Выравнивание по правому краю \*/*

  margin-bottom: 20px;

  padding: 10px 0;

  border-bottom: 1px solid #eee;

}

.sort-container label {

  margin-right: 10px;

  color: #555;

  font-weight: 500;

}

.sort-container select {

  padding: 8px 12px;

  border: 1px solid #d0d0d0;

  border-radius: 4px;

  background-color: #fff;

  font-size: 0.95em;

  cursor: pointer;

}

.pagination-nav {

  display: flex;

  justify-content: center;

  margin-top: 30px;

}

.pagination-nav button,

.pagination-nav .page-number {

  margin: 0 6px;

  padding: 8px 14px;

  border: 1px solid #d0d0d0;

  background-color: #fff;

  border-radius: 5px;

  cursor: pointer;

  font-size: 1em;

  color: #0056b3;

  transition: background-color 0.2s ease, border-color 0.2s ease, color 0.2s ease;

}

.pagination-nav button:hover:not(.active) {

  background-color: #eef;

  border-color: #0056b3;

}

.pagination-nav .page-number.active {

  background-color: #0056b3;

  color: white;

  border-color: #0056b3;

  cursor: default;

}

* + Стили .invalid и .error-text реализуют «визуальное уведомление» (красная рамка/фон и текст ошибки) для формы (ConceptForm).
  + .pagination-nav, .page-number.active — стили для постраничной навигации из пункта 3.
  + .sort-container — обёртка для селекта сортировки.

Обновляем App.js

import React, { useState } from 'react';

import ConceptList from './components/ConceptList';

import ConceptForm from './components/ConceptForm';

import './components/styles.css';

function **App**() {

  const [concepts, **setConcepts**] = useState([

    { id: 1, name: 'Алгоритмы', definition: 'Последовательность действий для решения задачи.', level: 3 },

    { id: 2, name: 'Структуры данных', definition: 'Способы организации данных для эффективного доступа и модификации.', level: 4 },

    { id: 3, name: 'Операционные системы', definition: 'Комплекс программ, управляющих ресурсами компьютера.', level: 5 },

    { id: 4, name: 'Компьютерные сети', definition: 'Совокупность компьютеров и устройств, соединенных для обмена данными.', level: 3 },

    { id: 5, name: 'Базы данных', definition: 'Организованная коллекция взаимосвязанных данных.', level: 4 },

    { id: 6, name: 'Машинное обучение', definition: 'Область ИИ, позволяющая системам обучаться на данных.', level: 5 },

  ]);

  const **handleAddConcept** = (*newConcept*) => {

    setConcepts((*prevConcepts*) => [...*prevConcepts*, *newConcept*]);

  };

  return (

    <div *className*="App">

      <h1 *style*={{ textAlign: 'center' }}>Список концепций компьютерных наук</h1>

      <ConceptForm *onAddConcept*={handleAddConcept} />

      <ConceptList *concepts*={concepts} />

    </div>

  );

}

export default App;

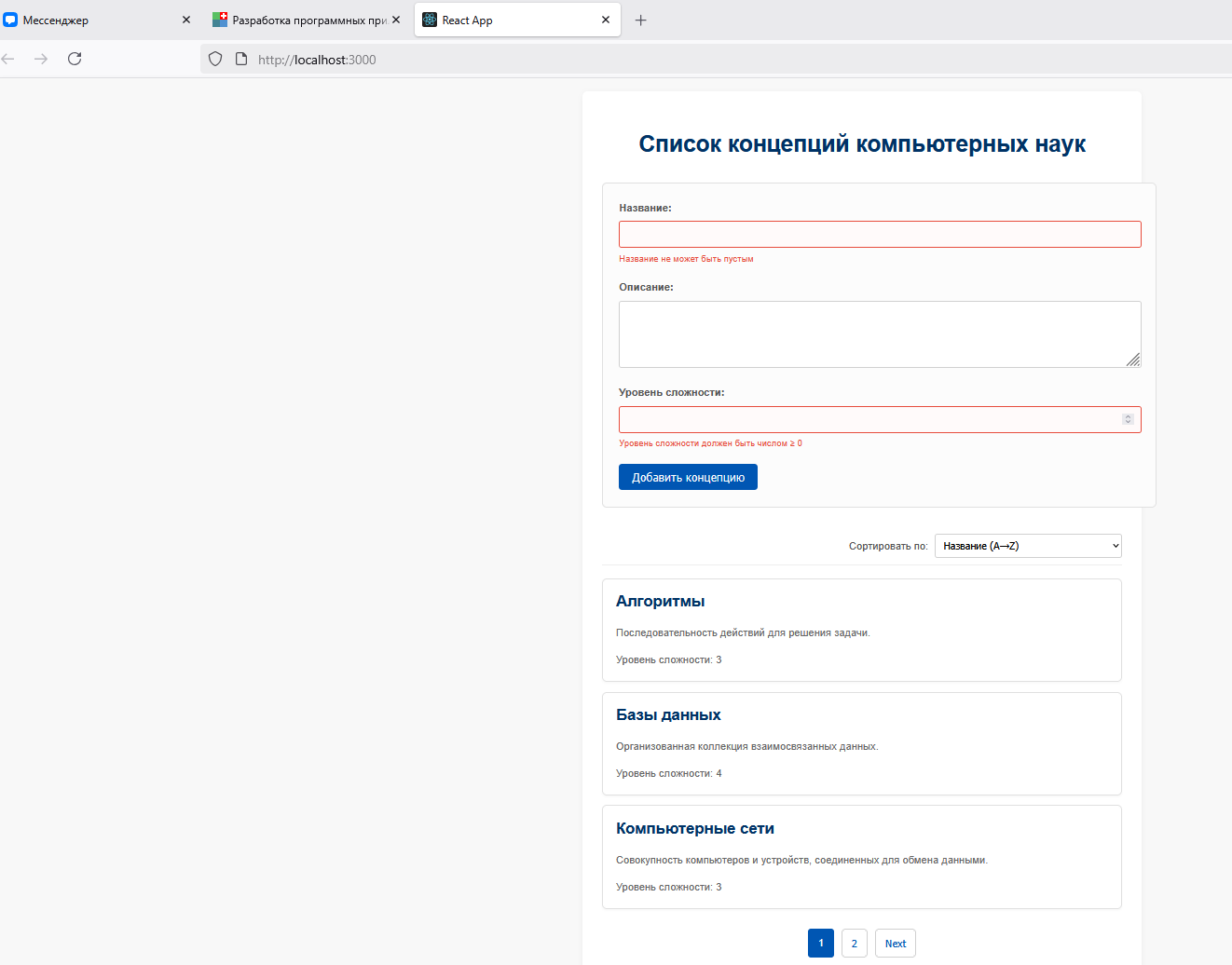
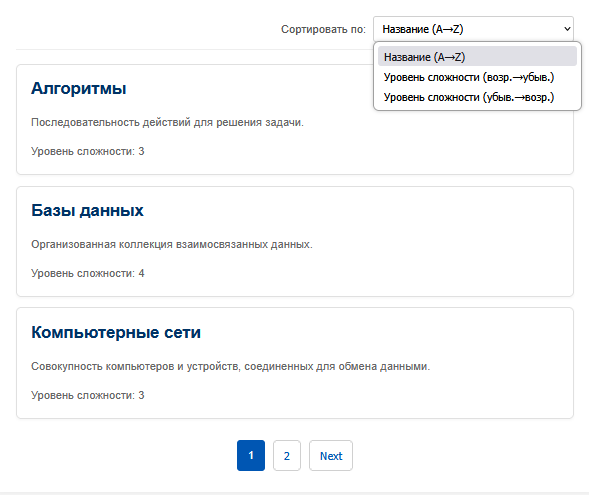


Рисунок 4 – Проверка заполнения.

  
Рисунок 5 – Сортировка.

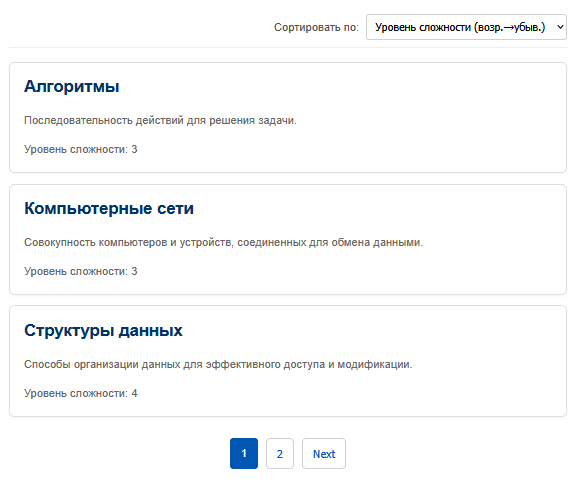


Рисунок 6 – Сортировка.

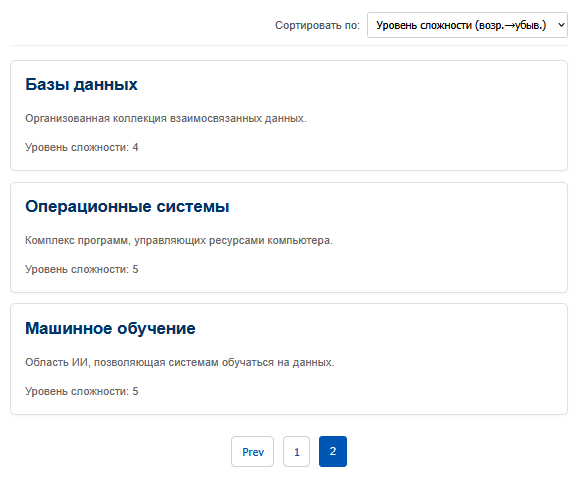


Рисунок 7 – Пагинация.

**Листинг программы.**

ConceptForm.js

import React, { useState } from 'react';

function **ConceptForm**({ *onAddConcept* }) {

  const [name, **setName**] = useState('');

  const [definition, **setDefinition**] = useState('');

  const [level, **setLevel**] = useState('');

  const [errors, **setErrors**] = useState({});

  const **validate** = () => {

    const errs = {};

    if (!name.trim()) errs.name = 'Название не может быть пустым';

    if (level === '' || isNaN(level) || Number(level) < 0) {

      errs.level = 'Уровень сложности должен быть числом ≥ 0';

    }

    return errs;

  };

  const **handleSubmit** = (*e*) => {

*e*.preventDefault();

    const validationErrors = validate();

    if (Object.keys(validationErrors).length > 0) {

      setErrors(validationErrors);

      return;

    }

    const newConcept = {

      id: Date.now(),

      name: name.trim(),

      definition: definition.trim(),

      level: Number(level),

    };

    onAddConcept(newConcept);

    setName('');

    setDefinition('');

    setLevel('');

    setErrors({});

  };

  return (

    <form *className*="concept-form" *onSubmit*={handleSubmit}>

      <div *className*="form-group">

        <label>Название:</label>

        <input

*type*="text"

*value*={name}

*onChange*={(*e*) => setName(*e*.target.value)}

*className*={errors.name ? 'invalid' : ''}

        />

        {errors.name && <div *className*="error-text">{errors.name}</div>}

      </div>

      <div *className*="form-group">

        <label>Описание:</label>

        <textarea

*value*={definition}

*onChange*={(*e*) => setDefinition(e.target.value)}

        />

      </div>

      <div *className*="form-group">

        <label>Уровень сложности:</label>

        <input

*type*="number"

*value*={level}

*onChange*={(*e*) => setLevel(e.target.value)}

*className*={errors.level ? 'invalid' : ''}

        />

        {errors.level && <div *className*="error-text">{errors.level}</div>}

      </div>

      <button *type*="submit" *className*="btn-submit">

        Добавить концепцию

      </button>

    </form>

  );

}

export default ConceptForm;

ConceptItem.js:

import React from 'react';

function **ConceptItem**({ *concept* }) {

  return (

    <li *className*="concept-item">

      <h3>{*concept*.name}</h3>

      <p>{*concept*.definition}</p>

      <p>Уровень сложности: {*concept*.level}</p>

    </li>

  );

}

export default ConceptItem;

ConceptList.js:

import React, { useState, useMemo } from 'react';

import ConceptItem from './ConceptItem';

function **ConceptList**({ *concepts* }) {

  const [sortKey, **setSortKey**] = useState('name');

  const sortedConcepts = useMemo(() => {

    let sorted = [...*concepts*];

    if (sortKey === 'name') {

      sorted.sort((*a*, *b*) => *a*.name.localeCompare(*b*.name));

    }

    else if (sortKey === 'levelAsc') {

      sorted.sort((*a*, *b*) => *a*.level - *b*.level);

    } else if (sortKey === 'levelDesc') {

      sorted.sort((*a*, *b*) => *b*.level - *a*.level);

    }

    return sorted;

  }, [*concepts*, sortKey]);

  const [currentPage, **setCurrentPage**] = useState(1);

  const itemsPerPage = 3;

  const totalPages = Math.ceil(sortedConcepts.length / itemsPerPage);

  const **getPageRange** = () => {

    const delta = 1;

    const range = [];

    for (

      let num = Math.max(1, currentPage - delta);

      num <= Math.min(totalPages, currentPage + delta);

      num++

    ) {

      range.push(num);

    }

    return range;

  };

  const startIdx = (currentPage - 1) \* itemsPerPage;

  const endIdx = startIdx + itemsPerPage;

  const conceptsToDisplay = sortedConcepts.slice(startIdx, endIdx);

  return (

    <div>

      <div *className*="sort-container">

        <label>Сортировать по: </label>

        <select *value*={sortKey} *onChange*={(*e*) => setSortKey(*e*.target.value)}>

          <option *value*="name">Название (A→Z)</option>

          <option *value*="levelAsc">Уровень сложности (возр.→убыв.)</option>

          <option *value*="levelDesc">Уровень сложности (убыв.→возр.)</option>

        </select>

      </div>

      <ul *className*="list-articles">

        {conceptsToDisplay.map((*concept*) => (

          <ConceptItem *key*={*concept*.id} *concept*={*concept*} />

        ))}

      </ul>

      {totalPages > 1 && (

        <nav *className*="pagination-nav">

          {currentPage > 1 && (

            <button *onClick*={() => setCurrentPage((*prev*) => prev - 1)}>

              Prev

            </button>

          )}

          {getPageRange().map((*num*) =>

            num === currentPage ? (

              <span *key*={num} *className*="page-number active">

                {num}

              </span>

            ) : (

              <button *key*={num} *onClick*={() => setCurrentPage(num)}>

                {num}

              </button>

            )

          )}

          {currentPage < totalPages && (

            <button *onClick*={() => setCurrentPage((*prev*) => prev + 1)}>

              Next

            </button>

          )}

        </nav>

      )}

    </div>

  );

}

export default ConceptList;

Styles.ccs:

body {

  font-family: 'Roboto', 'Helvetica Neue', Arial, sans-serif;

  line-height: 1.6;

  color: #333;

  background-color: #f8f8f8;

  margin: 0;

  padding: 20px;

}

.App {

  max-width: 800px;

  margin: 0 auto;

  background-color: #fff;

  padding: 30px;

  border-radius: 8px;

  box-shadow: 0 2px 10px rgba(0, 0, 0, 0.05);

}

h1 {

  color: #003366;

  font-size: 2.2em;

  margin-bottom: 30px;

  text-align: center;

}

.concept-form {

  width: 100%;

  margin-bottom: 30px;

  padding: 25px;

  border: 1px solid #e0e0e0;

  border-radius: 6px;

  background-color: #fcfcfc;

}

.form-group {

  margin-bottom: 20px;

}

.form-group label {

  display: block;

  font-weight: 600;

  margin-bottom: 8px;

  color: #555;

}

.form-group input,

.form-group textarea {

  width: calc(100% - 20px);

  padding: 10px;

  border: 1px solid #d0d0d0;

  border-radius: 4px;

  font-size: 1em;

}

.form-group textarea {

  min-height: 80px;

  resize: vertical;

}

.form-group input.invalid,

.form-group textarea.invalid {

  border-color: #e74c3c; */\* Красный для ошибок \*/*

  background-color: #fffafa;

}

.error-text {

  color: #e74c3c;

  font-size: 0.85em;

  margin-top: 5px;

}

.btn-submit {

  background-color: #0056b3; */\* Темно-синий \*/*

  color: #fff;

  border: none;

  padding: 10px 20px;

  border-radius: 5px;

  cursor: pointer;

  font-size: 1.1em;

  transition: background-color 0.2s ease;

}

.btn-submit:hover {

  background-color: #004085;

}

.list-articles {

  list-style: none;

  padding: 0;

  margin: 0;

}

.concept-item {

  border: 1px solid #e0e0e0;

  padding: 15px 20px;

  border-radius: 6px;

  margin-bottom: 15px;

  background-color: #fff;

  box-shadow: 0 1px 4px rgba(0, 0, 0, 0.03);

}

.concept-item h3 {

  color: #003366;

  margin-top: 0;

  margin-bottom: 8px;

  font-size: 1.5em;

}

.concept-item p {

  color: #666;

  margin-bottom: 5px;

}

.sort-container {

  display: flex;

  align-items: center;

  justify-content: flex-end; */\* Выравнивание по правому краю \*/*

  margin-bottom: 20px;

  padding: 10px 0;

  border-bottom: 1px solid #eee;

}

.sort-container label {

  margin-right: 10px;

  color: #555;

  font-weight: 500;

}

.sort-container select {

  padding: 8px 12px;

  border: 1px solid #d0d0d0;

  border-radius: 4px;

  background-color: #fff;

  font-size: 0.95em;

  cursor: pointer;

}

.pagination-nav {

  display: flex;

  justify-content: center;

  margin-top: 30px;

}

.pagination-nav button,

.pagination-nav .page-number {

  margin: 0 6px;

  padding: 8px 14px;

  border: 1px solid #d0d0d0;

  background-color: #fff;

  border-radius: 5px;

  cursor: pointer;

  font-size: 1em;

  color: #0056b3;

  transition: background-color 0.2s ease, border-color 0.2s ease, color 0.2s ease;

}

.pagination-nav button:hover:not(.active) {

  background-color: #eef;

  border-color: #0056b3;

}

.pagination-nav .page-number.active {

  background-color: #0056b3;

  color: white;

  border-color: #0056b3;

  cursor: default;

}

App.js:

import React, { useState } from 'react';

import ConceptList from './components/ConceptList';

import ConceptForm from './components/ConceptForm';

import './components/styles.css';

function **App**() {

  const [concepts, **setConcepts**] = useState([

    { id: 1, name: 'Алгоритмы', definition: 'Последовательность действий для решения задачи.', level: 3 },

    { id: 2, name: 'Структуры данных', definition: 'Способы организации данных для эффективного доступа и модификации.', level: 4 },

    { id: 3, name: 'Операционные системы', definition: 'Комплекс программ, управляющих ресурсами компьютера.', level: 5 },

    { id: 4, name: 'Компьютерные сети', definition: 'Совокупность компьютеров и устройств, соединенных для обмена данными.', level: 3 },

    { id: 5, name: 'Базы данных', definition: 'Организованная коллекция взаимосвязанных данных.', level: 4 },

    { id: 6, name: 'Машинное обучение', definition: 'Область ИИ, позволяющая системам обучаться на данных.', level: 5 },

  ]);

  const **handleAddConcept** = (*newConcept*) => {

    setConcepts((*prevConcepts*) => [...*prevConcepts*, *newConcept*]);

  };

  return (

    <div *className*="App">

      <h1 *style*={{ textAlign: 'center' }}>Список концепций компьютерных наук</h1>

      <ConceptForm *onAddConcept*={handleAddConcept} />

      <ConceptList *concepts*={concepts} />

    </div>

  );

}

export default App;

**Вывод**

В ходе выполнения лабораторной работы № 13 мы полностью перенесли функциональность предыдущих версий (лабораторных 5–12) на сторону клиента, используя современный стек React. Сначала убедились, что у нас установлены Node.js и npm, создали новое приложение с помощью Create React App и запустили его в dev-режиме без ошибок. Затем разобили пользовательский интерфейс на несколько независимых компонентов: форму для добавления новой записи и компонент списка концепций, каждый элемент которого оборачивается в отдельный компонент с уникальным ключом. Мы сохранили клиентские данные непосредственно в состоянии React, отказавшись от серверного бэкенда и базы данных, но сохранили весь привычный функционал отображения списка и добавления новых элементов.

Особое внимание было уделено сортировке и валидации. В компоненте списка реализован «умный» выбор параметра сортировки (по названию и по цене в обоих направлениях), а для избежания лишних перерисовок мы передаём React-компонентам поле key={id} и вычисляем отсортированный массив через useMemo. В форме добавления мы проверяем, что название не пустое, а цена — неотрицательное число; при ошибке поле автоматически получает CSS-класс .invalid, меняющий цвет рамки и фона, а под ним выводится текст ошибки. Как только все проверки пройдены, новый объект «концепция» добавляется в состояние верхнего компонента, и список автоматически перерисовывается.

В результате у нас получилась полноценная SPA-страница, где можно:

— добавлять новые «концепции» без перезагрузки страницы,

— сортировать записи по нескольким признакам,

— мгновенно видеть подсветку ошибок в форме,

— при этом React берёт на себя минимизацию ненужных обновлений через корректную работу с key и useMemo.

Таким образом, все пункты задания выполнены: настроены инструменты разработки, создана аналогичная страница на React, добавлена клиентская сортировка с оптимизацией, а также реализована интерактивная форма с валидацией и визуальным оповещением об ошибках.