ДИСЦИПЛИНА	Фронтенд и бэкенд разработка
ИНСТИТУТ	ИПТИП
КАФЕДРА	Индустриального программирования
ВИД УЧЕБНОГО МАТЕРИАЛА	Методические указания к практическим занятиям по дисциплине
ПРЕПОДАВАТЕЛЬ	Загородних Николай Анатольевич Краснослободцева Дарья Борисовна
СЕМЕСТР	4 семестр, 2025/2026 уч. год

Практическое занятие 4

API + React

Подробнее рассмотрим CRUD операции (Create, Read, Update, Delete) в API и свяжем его с интерфейсом на React. Решение практического задания осуществляется внутри соответствующей рабочей тетради, расположенной в СДО.

Использование API на клиенте

Для того, чтобы связать клиентскую часть вашего приложения с серверной, необходимо обращаться к маршрутам вашего API в коде клиента, используя для этого специальные средства, представляющие из себя отдельные функции или целые библиотеки.

Среди популярных средств с таким функционалом можно выделить 2: `fetch` - функция, встроенная в JavaScript, которая используется для отправки HTTP-запросов, а также `axios` - библиотека, предоставляющая широкий функционал по взаимодействию с HTTP-запросами и обработке их ответов.

В рамках практического занятия создадим два небольших приложение на React и Express (для клиента и сервера соответственно), после чего свяжем их в единую систему - сайт.

Создание бэкенда

Установим зависимости: `npm i express nanoid`

Дополним код из предыдущих практических занятий:

```
const express = require('express');
const { nanoid } = require('nanoid');
```

```
const app = express();
const port = 3000;

let users = [
    {id: nanoid(6), name: 'Петр', age: 16},
    {id: nanoid(6), name: 'Иван', age: 18},
    {id: nanoid(6), name: 'Дарья', age: 20},
]

// Middleware для парсинга JSON
app.use(express.json());

// Middleware для логирования запросов
app.use((req, res, next) => {
    res.on('finish', () => {
        console.log(`[${new Date().toISOString()}] [${req.method}]
${res.statusCode} ${req.path}`);
        if (req.method === 'POST' || req.method === 'PUT' || req.method ===
'PATCH') {
            console.log('Body:', req.body);
        }
    });
    next();
});

// Функция-помощник для получения пользователя из списка
function findUserOr404(id, res) {
    const user = users.find(u => u.id == id);
    if (!user) {
        res.status(404).json({ error: "User not found" });
        return null;
    }
    return user;
}

// Функция-помощник

// POST /api/users
app.post("/api/users", (req, res) => {
    const { name, age } = req.body;

    const newUser = {
        id: nanoid(6),
        name: name.trim(),
        age: Number(age),
```

```
    };

    users.push(newUser);
    res.status(201).json(newUser);
});

// GET /api/users
app.get("/api/users", (req, res) => {
    res.json(users);
});

// GET /api/users/:id
app.get("/api/users/:id", (req, res) => {
    const id = req.params.id;

    const user = findUserOr404(id, res);
    if (!user) return;

    res.json(user);
});

// PATCH /api/users/:id
app.patch("/api/users/:id", (req, res) => {
    const id = req.params.id;

    const user = findUserOr404(id, res);
    if (!user) return;

    // Нельзя PATCH без полей
    if (req.body?.name === undefined && req.body?.age === undefined) {
        return res.status(400).json({
            error: "Nothing to update",
        });
    }

    const { name, age } = req.body;

    if (name !== undefined) user.name = name.trim();
    if (age !== undefined) user.age = Number(age);

    res.json(user);
});

// DELETE /api/users/:id
app.delete("/api/users/:id", (req, res) => {
    const id = req.params.id;
```

```
    const exists = users.some((u) => u.id === id);
    if (!exists) return res.status(404).json({ error: "User not found" });

    users = users.filter((u) => u.id !== id);

    // Правильнее 204 без тела
    res.status(204).send();
});

// 404 для всех остальных маршрутов
app.use((req, res) => {
    res.status(404).json({ error: "Not found" });
});

// Глобальный обработчик ошибок (чтобы сервер не падал)
app.use((err, req, res, next) => {
    console.error("Unhandled error:", err);
    res.status(500).json({ error: "Internal server error" });
});

// Запуск сервера
app.listen(port, () => {
    console.log(`Сервер запущен на http://localhost:${port}`);
});
```

Основные изменения:

1. Добавили middleware для логирования запросов

```
app.use((req, res, next) => {
    res.on('finish', () => {
        console.log(`[${new Date().toISOString()}] [${req.method}]
${res.statusCode} ${req.path}`);
        if (req.method === 'POST' || req.method === 'PUT' || req.method ===
'PATCH') {
            console.log('Body:', req.body);
        }
    });
    next();
});
```

2. Добавили обработчики ошибок и проверки корректности данных в запросах

```
// 404 для всех остальных маршрутов
app.use((req, res) => {
    res.status(404).json({ error: "Not found" });
});

// Глобальный обработчик ошибок (чтобы сервер не падал)
app.use((err, req, res, next) => {
    console.error("Unhandled error:", err);
    res.status(500).json({ error: "Internal server error" });
});
```

3. Заменили численные идентификаторы на `nanoid` - аналог uuid, позволяющий генерировать случайные строки определённой длины

```
const { nanoid } = require("nanoid");

nanoid(6)  // Генерация ID с длиной 6 символов
```

4. Добавили функцию `findUserOr404()` для удобства нахождения пользователя по id

```
function findUserOr404(id, res) {
    const user = users.find(u => u.id == id);
    if (!user) {
        res.status(404).json({ error: "User not found" });
        return null;
    }
    return user;
}
```

Итак, наш сервер содержит следующие маршруты:

Путь	Метод	Описание
/api/users	POST	Создание нового пользователя
/api/users	GET	Получение списка пользователей
/api/users/:id	GET	Получение пользователя по ID
/api/users/:id	PATCH	Изменение пользователя
/api/users/:id	DELETE	Удаление пользователя

Создание фронтенда

Определимся со структурой React-приложения. Мы будем использовать React и Sass, т.к. приложение будет небольшим, ограничимся следующими файлами и директориями:

```
public/
├─ css/
│  └─ UsersPage.css
└─ index.html
src/
├─ api/
│  └─ index.js
├─ components/
│  ├─ UserItem.jsx
│  ├─ UserModal.jsx
│  └─ UsersList.jsx
├─ pages/
│  └─ UsersPage/
│     ├─ UsersPage.jsx
│     └─ UsersPage.scss
├─ App.js
├─ index.css
└─ index.js
package.json
package-lock.json
```

Приступим к созданию и наполнению файлов кодом.

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8"/>
    <link href="%PUBLIC_URL%/favicon.ico" rel="icon"/>
    <meta content="width=device-width, initial-scale=1" name="viewport"/>
    <title>React Users</title>
</head>
<body>
    <div id="root"></div>
</body>
</html>
```

index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(
    <React.StrictMode>
        <App/>
    </React.StrictMode>
);
```

index.css

```
body {
    margin: 0;
    font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto',
'Oxygen',
    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
    sans-serif;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}

code {
    font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',
    monospace;
}
```

App.js

```
import UsersPage from "./pages/UsersPage/UsersPage";

function App() {
    return (
        <div className="App">
            <UsersPage />
        </div>
    );
}

export default App;
```

UsersPage.jsx

```jsx
import React, { useMemo, useState } from "react";
import "./UsersPage.css";

import UsersList from "../../components/UsersList";
import UserModal from "../../components/UserModal";

export default function UsersPage() {
    const [users, setUsers] = useState([
        { id: 1, name: "Петр", age: 16 },
        { id: 2, name: "Иван", age: 18 },
        { id: 3, name: "Дарья", age: 20 },
    ]);

    const [modalOpen, setModalOpen] = useState(false);
    const [modalMode, setModalMode] = useState("create"); // "create" | "edit"
    const [editingUser, setEditingUser] = useState(null);

    const nextId = useMemo(() => {
        const maxId = users.reduce((m, u) => Math.max(m, u.id), 0);
        return maxId + 1;
    }, [users]);

    const openCreate = () => {
        setModalMode("create");
        setEditingUser(null);
        setModalOpen(true);
    };

    const openEdit = (user) => {
        setModalMode("edit");
        setEditingUser(user);
        setModalOpen(true);
    };

    const closeModal = () => {
        setModalOpen(false);
        setEditingUser(null);
    };

    const handleDelete = (id) => {
        const ok = window.confirm("Удалить пользователя?");
        if (!ok) return;
        setUsers((prev) => prev.filter((u) => u.id !== id));
    };
```

```
    const handleSubmitModal = (payload) => {
        if (modalMode === "create") {
            setUsers((prev) => [...prev, { id: nextId, name: payload.name,
age: payload.age }]);
        } else {
            setUsers((prev) =>
                prev.map((u) => (u.id === payload.id ? {  ...u, name:
payload.name, age: payload.age } : u))
            );
        }
        closeModal();
    };

    return (
        <div className="page">
            <header className="header">
                <div className="header__inner">
                    <div className="brand">Users App</div>
                    <div className="header__right">React</div>
                </div>
            </header>

            <main className="main">
                <div className="container">
                    <div className="toolbar">
                        <h1 className="title">Пользователи</h1>
                        <button className="btn btn--primary" onClick=
{openCreate}>
                            + Создать
                        </button>
                    </div>

                    <UsersList users={users} onEdit={openEdit} onDelete=
{handleDelete} />
                </div>
            </main>

            <footer className="footer">
                <div className="footer__inner">© {new Date().getFullYear()}
Users App</div>
            </footer>

            <UserModal
                open={modalOpen}
                mode={modalMode}
```

```
                initialUser={editingUser}
                onClose={closeModal}
                onSubmit={handleSubmitModal}
            />
        </div>
    );
}
```

UsersList.jsx

```
import React from "react";
import UserItem from "./UserItem";

export default function UsersList({ users, onEdit, onDelete }) {
    if (!users.length) {
        return <div className="empty">Пользователей пока нет</div>;
    }

    return (
        <div className="list">
            {users.map((u) => (
                <UserItem key={u.id} user={u} onEdit={onEdit} onDelete=
{onDelete} />
            ))}
        </div>
    );
}
```

UserItem.jsx

```
import React from "react";

export default function UserItem({ user, onEdit, onDelete }) {
    return (
        <div className="userRow">
            <div className="userMain">
                <div className="userId">#{user.id}</div>
                <div className="userName">{user.name}</div>
                <div className="userAge">{user.age} лет</div>
            </div>

            <div className="userActions">
                <button className="btn" onClick={() => onEdit(user)}>
                    Редактировать
                </button>
```

```
                <button className="btn btn--danger" onClick={() =>
onDelete(user.id)}>
                    Удалить
                </button>
            </div>
        </div>
    );
}
```

UserModal.jsx

```
import React, { useEffect, useState } from "react";

export default function UserModal({ open, mode, initialUser, onClose, onSubmit
}) {
    const [name, setName] = useState("");
    const [age, setAge] = useState("");

    useEffect(() => {
        if (!open) return;
        setName(initialUser?.name ?? "");
        setAge(initialUser?.age != null ? String(initialUser.age) : "");
    }, [open, initialUser]);

    if (!open) return null;

    const title = mode === "edit" ? "Редактирование пользователя" : "Создание
пользователя";

    const handleSubmit = (e) => {
        e.preventDefault();

        const trimmed = name.trim();
        const parsedAge = Number(age);

        if (!trimmed) {
            alert("Введите имя");
            return;
        }
        if (!Number.isFinite(parsedAge) || parsedAge < 0 || parsedAge > 150) {
            alert("Введите корректный возраст (0–150)");
            return;
        }

        onSubmit({
            id: initialUser?.id,
```

```
            name: trimmed,
            age: parsedAge,
        });
    };

    return (
        <div className="backdrop" onMouseDown={onClose}>
            <div className="modal" onMouseDown={(e) => e.stopPropagation()}
role="dialog" aria-modal="true">
                <div className="modal__header">
                    <div className="modal__title">{title}</div>
                    <button className="iconBtn" onClick={onClose} aria-
label="Закрыть">
                        ×
                    </button>
                </div>

                <form className="form" onSubmit={handleSubmit}>
                    <label className="label">
                        Имя
                        <input
                            className="input"
                            value={name}
                            onChange={(e) => setName(e.target.value)}
                            placeholder="Например, Иван"
                            autoFocus
                        />
                    </label>

                    <label className="label">
                        Возраст
                        <input
                            className="input"
                            value={age}
                            onChange={(e) => setAge(e.target.value)}
                            placeholder="Например, 20"
                            inputMode="numeric"
                        />
                    </label>

                    <div className="modal__footer">
                        <button type="button" className="btn" onClick=
{onClose}>
                            Отмена
                        </button>
                        <button type="submit" className="btn btn--primary">
```

```
                            {mode === "edit" ? "Сохранить" : "Создать"}
                        </button>
                    </div>
                </form>
            </div>
        </div>
    );
}
```

Стилизуем страницу

Добавим файл UsersPage.scss:

```scss
$bg: #0b0f19;
$panel: rgba(255, 255, 255, 0.03);
$border: rgba(255, 255, 255, 0.10);
$text: #e7eaf3;

$primary: rgba(99, 102, 241, 0.55);
$primaryBg: rgba(99, 102, 241, 0.18);

$danger: rgba(239, 68, 68, 0.45);
$dangerBg: rgba(239, 68, 68, 0.12);

@mixin card($radius: 12px) {
  border-radius: $radius;
  border: 1px solid $border;
  background: $panel;
}

@mixin container($w: 900px) {
  max-width: $w;
  width: 100%;
  margin: 0 auto;
}

.page {
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  background: $bg;
  color: $text;
  font-family: system-ui, -apple-system, Segoe UI, Roboto, Arial;
}

.header {
```

```scss
  border-bottom: 1px solid rgba(255, 255, 255, 0.08);
  background: rgba(255, 255, 255, 0.02);

  &__inner {
    @include container();
    padding: 14px 16px;
    display: flex;
    align-items: center;
    justify-content: space-between;
  }

  &__right {
    opacity: 0.75;
    font-size: 13px;
  }
}

.brand {
  font-weight: 700;
  letter-spacing: 0.3px;
}

.main {
  flex: 1;
  display: flex;
}

.container {
  @include container();
  padding: 16px;
}

.toolbar {
  display: flex;
  align-items: center;
  justify-content: space-between;
  gap: 12px;
  margin-bottom: 12px;

  .title {
    margin: 0;
    font-size: 20px;
  }
}

// ===== Buttons =====
```

```scss
.btn {
  padding: 8px 10px;
  border-radius: 10px;
  border: 1px solid rgba(255, 255, 255, 0.14);
  background: transparent;
  color: inherit;
  cursor: pointer;

  &--primary {
    background: $primaryBg;
    border: 1px solid $primary;
  }

  &--danger {
    background: $dangerBg;
    border: 1px solid $danger;
  }
}

.list {
  display: flex;
  flex-direction: column;
  gap: 10px;
}

.empty {
  padding: 18px;
  border-radius: 12px;
  border: 1px dashed rgba(255, 255, 255, 0.18);
  opacity: 0.85;
}

.userRow {
  @include card(12px);
  display: flex;
  align-items: center;
  justify-content: space-between;
  gap: 12px;
  padding: 12px;
}

.userMain {
  display: flex;
  align-items: center;
  gap: 12px;
  flex-wrap: wrap;
```

```scss
  .userId {
    opacity: 0.7;
    font-variant-numeric: tabular-nums;
  }

  .userName {
    font-weight: 600;
  }

  .userAge {
    opacity: 0.85;
  }
}

.userActions {
  display: flex;
  gap: 8px;
}

.footer {
  border-top: 1px solid rgba(255, 255, 255, 0.08);
  background: rgba(255, 255, 255, 0.02);

  &__inner {
    @include container();
    padding: 14px 16px;
    opacity: 0.7;
    font-size: 13px;
  }
}

.backdrop {
  position: fixed;
  inset: 0;
  background: rgba(0, 0, 0, 0.55);
  display: flex;
  align-items: center;
  justify-content: center;
  padding: 16px;
}

.modal {
  width: 100%;
  max-width: 420px;
  border-radius: 16px;
```

```scss
  border: 1px solid rgba(255, 255, 255, 0.12);
  background: #0f1526;
  box-shadow: 0 20px 60px rgba(0, 0, 0, 0.35);
  overflow: hidden;

  &__header {
    padding: 12px;
    border-bottom: 1px solid $border;
    display: flex;
    align-items: center;
    justify-content: space-between;
  }

  &__title {
    font-weight: 700;
  }

  &__footer {
    display: flex;
    justify-content: flex-end;
    gap: 8px;
    margin-top: 4px;
  }
}

.iconBtn {
  width: 34px;
  height: 34px;
  border-radius: 10px;
  border: 1px solid rgba(255, 255, 255, 0.12);
  background: transparent;
  color: inherit;
  cursor: pointer;
}

.form {
  padding: 12px;
  display: flex;
  flex-direction: column;
  gap: 10px;
}

.label {
  display: flex;
  flex-direction: column;
  gap: 6px;
```

```
  font-size: 13px;
  opacity: 0.95;
}

.input {
  padding: 10px 12px;
  border-radius: 12px;
  border: 1px solid rgba(255, 255, 255, 0.12);
  background: rgba(255, 255, 255, 0.04);
  color: inherit;
  outline: none;
}
```

Теперь для автоматической генерации css из sass добавим следующую конфигурацию в package.json:

```
"scripts": {
    "start": "set PORT=3001 && react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject",
    "sass": "sass src --style compressed --quiet",
    "dev": "npm run sass & npm run start"
},
```

Таким образом, команда `npm run dev` запустит сервер и сгенерирует нужный нам css-файл.

Кроме того, в скрипте `start` приведён пример изменения порта для запуска фронтенда.

Подключение бэкенда к фронтенду

Теперь у нас есть полностью рабочий фронтенд и полностью рабочий бэкенд, осталось связать это в одно целое.

На фронтенде установим axios: `npm i axios`

Создадим файл `src/api/index.js`

```
import axios from "axios";

const apiClient = axios.create({
    baseURL: "http://localhost:3000/api",
    headers: {
        "Content-Type": "application/json",
```

```
        "accept": "application/json",
    }
});

export const api = {

    createUser: async (user) => {
        let response = await apiClient.post("/users", user);
        return response.data;
    },

    getUsers: async () => {
        let response = await apiClient.get("/users");
        return response.data;
    },

    getUserById: async (id) => {
        let response = await apiClient.get(`/users/${id}`);
        return response.data;
    },

    updateUser: async (id, user) => {
        let response = await apiClient.patch(`/users/${id}`, user);
        return response.data;
    },

    deleteUser: async (id) => {
        let response = await apiClient.delete(`/users/${id}`);
        return response.data;
    }

}
```

Здесь мы создали API-клиента, которому задали базовый URL для удобства и автоматически устанавливающиеся заголовки во избежание некоторых ошибок в части обработки контента запросов:

```
const apiClient = axios.create({
    baseURL: "http://localhost:3000/api",
    headers: {
        "Content-Type": "application/json",
        "accept": "application/json",
    }
});
```

Теперь у нас есть константа `api`, через которую мы можем обращаться к любому маршруту нашего бэкенд-приложения, просто вызвав соответствующую функцию.

Дополним страницу пользователей

UsersPage.jsx

```
import React, { useEffect, useState } from "react";
import "./UsersPage.scss";

import UsersList from "../../components/UsersList";
import UserModal from "../../components/UserModal";
import { api } from "../../api";

export default function UsersPage() {
    const [users, setUsers] = useState([]);
    const [loading, setLoading] = useState(true);

    const [modalOpen, setModalOpen] = useState(false);
    const [modalMode, setModalMode] = useState("create"); // create | edit
    const [editingUser, setEditingUser] = useState(null);

    useEffect(() => {
        loadUsers();
    }, []);

    const loadUsers = async () => {
        try {
            setLoading(true);
            const data = await api.getUsers();
            setUsers(data);
        } catch (err) {
            console.error(err);
            alert("Ошибка загрузки пользователей");
        } finally {
            setLoading(false);
        }
    };

    const openCreate = () => {
        setModalMode("create");
        setEditingUser(null);
        setModalOpen(true);
    };

    const openEdit = (user) => {
```

```
        setModalMode("edit");
        setEditingUser(user);
        setModalOpen(true);
    };

    const closeModal = () => {
        setModalOpen(false);
        setEditingUser(null);
    };

    const handleDelete = async (id) => {
        const ok = window.confirm("Удалить пользователя?");
        if (!ok) return;

        try {
            await api.deleteUser(id);
            setUsers((prev) => prev.filter((u) => u.id !== id));
        } catch (err) {
            console.error(err);
            alert("Ошибка удаления пользователя");
        }
    };

    const handleSubmitModal = async (payload) => {
        try {
            if (modalMode === "create") {
                const newUser = await api.createUser(payload);
                setUsers((prev) => [...prev, newUser]);
            } else {
                const updatedUser = await api.updateUser(payload.id, payload);
                setUsers((prev) =>
                    prev.map((u) => (u.id === payload.id ? updatedUser : u))
                );
            }
            closeModal();
        } catch (err) {
            console.error(err);
            alert("Ошибка сохранения пользователя");
        }
    };

    return (
        <div className="page">
            <header className="header">
                <div className="header__inner">
                    <div className="brand">Users App</div>
```

```
                    <div className="header__right">React</div>
                </div>            </header>
            <main className="main">
                <div className="container">
                    <div className="toolbar">
                        <h1 className="title">Пользователи</h1>
                        <button className="btn btn--primary" onClick=
{openCreate}>
                            + Создать
                        </button>
                    </div>
                    {loading ? (
                        <div className="empty">Загрузка...</div>
                    ) : (
                        <UsersList
                            users={users}
                            onEdit={openEdit}
                            onDelete={handleDelete}
                        />
                    )}
                </div>
            </main>
            <footer className="footer">
                <div className="footer__inner">
                    © {new Date().getFullYear()} Users App
                </div>
            </footer>
            <UserModal                open={modalOpen}
                mode={modalMode}
                initialUser={editingUser}
                onClose={closeModal}
                onSubmit={handleSubmitModal}
            />
        </div>    );
}
```

Отлично, теперь все методы используются прямо в коде нашей страницы. Запустим клиента и сервера и перейдём в браузер.

Скорее всего вы увидите ошибку такого вида:

```
Access to XMLHttpRequest at 'http://localhost:3000/api/users' from origin
'http://localhost:3001' has been blocked by CORS policy: No 'Access-Control-
Allow-Origin' header is present on the requested resource.
```

Дело в том, что для обращения к маршрутам API из браузера, требуется разрешение политики CORS. Модифицируем бэкенд.

Установим соответствующую библиотеку: `npm i cors`

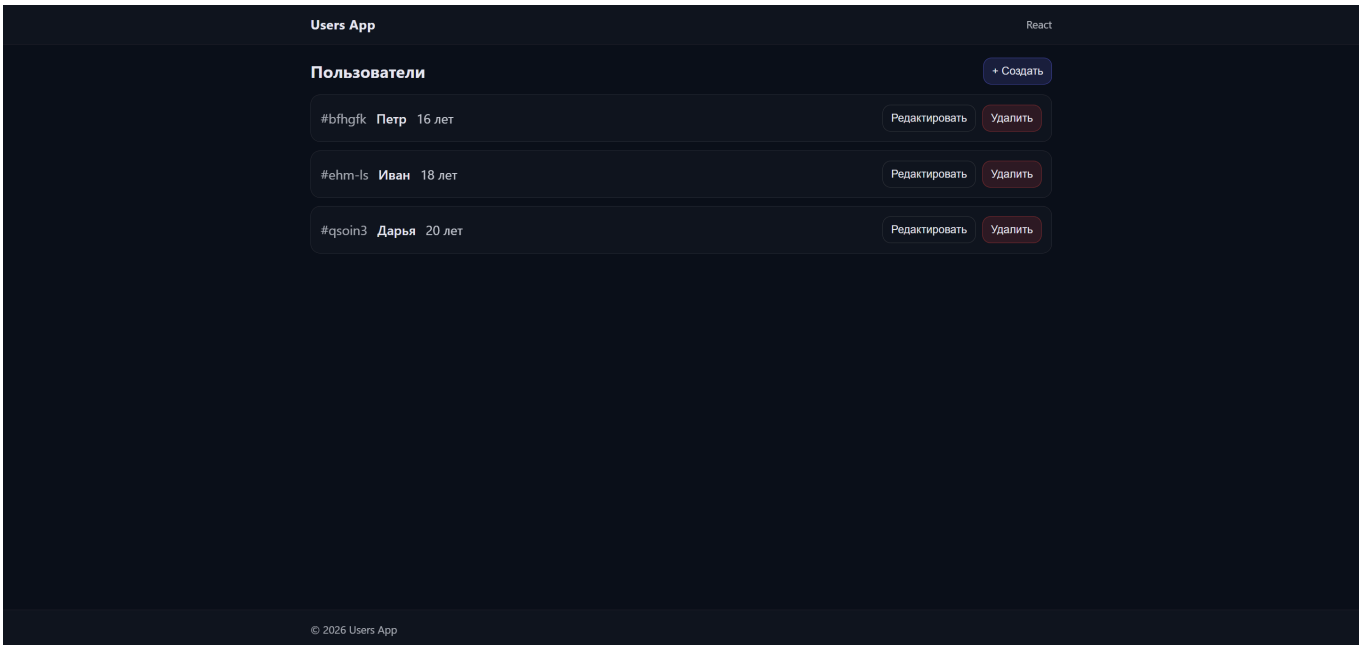
И добавим такую запись в app.js:

```
const cors = require("cors");

app.use(cors({
  origin: "http://localhost:3001",
  methods: ["GET", "POST", "PATCH", "DELETE"],
  allowedHeaders: ["Content-Type", "Authorization"],
}));
```

Теперь мы разрешили все запросы от клиента с адресом `http://localhost:3001` с методами GET, POST, PATCH, DELETE и заголовками Content-Type и Authorization.

Перезапустим бэкенд и снова перейдём в браузер - всё должно работать.



Практичкеское задание

Необходимо доработать рассмотренный пример: два небольших приложение на React и Express (для клиента и сервера, соответственно), связанных в единую систему - сайт интернет-магазина товаров с тематикой на ваше усмотрение. Количество товаров - не менее 10. Карточка товара должна содержать как минимум: название, категорию, описание товара, цену, количество на складе. Опционально доработать рейтинг и фото.

Формат отчета

В качестве ответа на задание необходимо прикрепить ссылку на репозиторий с реализованной практикой. Ссылка подгружается в соответствующий раздел СДО: Задания для самостоятельной работы -> Практические занятия 3-4.