Лабораторная работа 3: Разработка SPA с использованием React

Студент: Дмитриев Андрей Иванович
Группа: K3439
Вариант: Платформа для образовательных курсов и управления учебным процессом

---

1. Описание задачи

В рамках третьей лабораторной работы необходимо было мигрировать приложение на современный фреймворк React, реализовав:

- Маршрутизацию с помощью React Router
- Компонентный подход с переиспользуемыми компонентами
- Управление состоянием через Context API
- Custom hooks для выделения переиспользуемой логики
- Работу с внешним API через Axios

---

2. Архитектура приложения

2.1 Структура проекта

```
front_it_school/
├── public/
│   ├── index.html        # Точка входа HTML
│   ├── manifest.json     # PWA манифест
│   └── robots.txt        # SEO конфигурация
├── src/
│   ├── api/
│   │   └── axiosInstance.js        # Настроенный Axios клиент
│   ├── components/                 # Переиспользуемые компоненты
│   │   ├── Sidebar.jsx
```

```
│   │   ├── TopBar.jsx
│   │   ├── SmartTopBar.jsx
│   │   ├── CourseCard.jsx
│   │   ├── CourseImage.jsx
│   │   ├── CourseProgressBar.jsx
│   │   ├── Schedule.jsx
│   │   ├── Calendar.jsx
│   │   ├── BestCoin.jsx
│   │   ├── NewsModal.jsx
│   │   ├── EventModal.jsx
│   │   ├── ProductModal.jsx
│   │   ├── ConfirmModal.jsx
│   │   ├── Toast.jsx
│   │   └── ... (30+ компонентов)
│   ├── contexts/              # Context API
│   │   ├── AuthContext.js
│   │   └── NotificationsContext.js
│   ├── hooks/                 # Custom hooks
│   │   ├── useConfirm.js
│   │   └── useMobileKeyboard.js
│   ├── pages/                 # Страницы (роуты)
│   │   ├── HomePage.jsx
│   │   ├── LoginPage.jsx
│   │   ├── ProfilePage.jsx
│   │   ├── StudentCoursesPage.jsx
```

```
│   │   ├── StudentCoursePage.jsx
│   │   ├── StudentLessonPage.jsx
│   │   ├── TeacherCoursesPage.jsx
│   │   ├── ManageCoursePage.jsx
│   │   ├── ShopPage.jsx
│   │   ├── RatingPage.jsx
│   │   └── ... (35 страниц)
│   ├── services/              # API сервисы
│   │   ├── authService.js
│   │   ├── userService.js
│   │   ├── courseService.js
│   │   ├── lessonService.js
│   │   ├── homeworkService.js
│   │   ├── scheduleService.js
│   │   ├── productService.js
│   │   ├── notificationService.js
│   │   └── ... (20 сервисов)
│   ├── styles/                # CSS модули
│   │   ├── HomePage.css
│   │   ├── LoginPage.css
│   │   ├── CourseCard.css
│   │   └── ... (50+ стилей)
│   ├── App.js                 # Главный компонент с роутингом
│   └── index.js               # Точка входа React
├── package.json               # Зависимости
```

└── craco.config.js # Конфигурация сборки

---

3. Маршрутизация (React Router)

3.1 Настройка роутера (App.js)

```
import React from 'react';
import { BrowserRouter, Routes, Route, Navigate } from 'react-router-dom';

export default function App() {
  return (
    <NotificationsProvider>
      <BrowserRouter>
        <Routes>
          {/* Публичные маршруты */}
          <Route path="/login" element={<LoginPage />} />
          <Route path="/forgot-password" element={<ForgotPasswordPage />} />
          <Route path="/reset-password" element={<ResetPasswordPage />} />

          {/* Базовые приватные маршруты */}
          <Route path="/home" element={<PrivateRoute><HomePage /></PrivateRoute>} />
          <Route path="/profile" element={<PrivateRoute><ProfilePage /></PrivateRoute>} />
          <Route path="/coin-history" element={<PrivateRoute><CoinHistoryPage
/></PrivateRoute>} />

          {/* Маршруты студента */}
          <Route path="/courses" element={<PrivateRoute><StudentCoursesPage
/></PrivateRoute>} />
```

```
        <Route path="/courses/:courseId/student" element={<PrivateRoute><StudentCoursePage /></PrivateRoute>} />

        <Route path="/courses/:courseId/lessons/:lessonId" element={<PrivateRoute><StudentLessonPage /></PrivateRoute>} />

        <Route path="/rating" element={<PrivateRoute><RatingPage /></PrivateRoute>} />

        <Route path="/shop" element={<PrivateRoute><ShopPage /></PrivateRoute>} />


        {/* Маршруты преподавателя */}

        <Route path="/teacher-courses" element={<PrivateRoute><TeacherCoursesPage /></PrivateRoute>} />

        <Route path="/courses/:courseId/teacher" element={<PrivateRoute><TeacherCoursePage /></PrivateRoute>} />

        <Route path="/courses/:courseId/teacher/lessons/:lessonId" element={<PrivateRoute><TeacherLessonPage /></PrivateRoute>} />


        {/* Маршруты администратора */}

        <Route path="/manage/groups" element={<PrivateRoute><ManageGroupPage /></PrivateRoute>} />

        <Route path="/manage/students" element={<PrivateRoute><ManageStudentsPage /></PrivateRoute>} />

        <Route path="/manage/teachers" element={<PrivateRoute><ManageTeachersPage /></PrivateRoute>} />

        <Route path="/manage/courses" element={<PrivateRoute><ManageCoursePage /></PrivateRoute>} />

        <Route path="/manage/events" element={<PrivateRoute><ManageEventsPage /></PrivateRoute>} />

        <Route path="/manage/news" element={<PrivateRoute><ManageNewsPage /></PrivateRoute>} />
```

```
        {/* Редирект по умолчанию */}
        <Route path="*" element={<Navigate to="/home" />} />
      </Routes>
    </BrowserRouter>
  </NotificationsProvider>
 );
}
```

Особенности:

- Использование React Router v7
- Вложенные маршруты с параметрами (:courseId, :lessonId)
- Защищённые маршруты через компонент PrivateRoute
- Разделение маршрутов по ролям (student/teacher/admin)

3.2 Защищённые маршруты (Route Guards)

```
function PrivateRoute({ children }) {
 const { user } = useAuth();

 // Если user === undefined, значит ещё идёт загрузка
 if (user === undefined) {
  return (
   <div style={{ padding: '20px', textAlign: 'center' }}>
    Загрузка...
   </div>
  );
 }
```

```
  // Если user === null, значит не авторизован
  if (user === null) {
    return <Navigate to="/login" />;
  }

  // Если user есть, показываем контент
  return children;
}
```

Три состояния:

- undefined — идёт проверка сессии
- null — пользователь не авторизован → редирект на /login
- object — пользователь авторизован → показываем контент

3.3 Навигация и параметры маршрута

```
import { useNavigate, useParams } from 'react-router-dom';

export default function StudentLessonPage() {
  const navigate = useNavigate();
  const { courseId, lessonId } = useParams();

  const goBack = () => {
    navigate(`/courses/${courseId}/student`);
  };

  const goToNextLesson = (nextLessonId) => {
    navigate(`/courses/${courseId}/lessons/${nextLessonId}`);
```

```
  };


  // ...
}
```

Используемые хуки:

- useNavigate() — программная навигация
- useParams() — получение параметров из URL
- useLocation() — получение текущего location
- useSearchParams() — работа с query параметрами

---

4. Компонентный подход

4.1 Переиспользуемые компоненты

4.1.1 Sidebar — боковое меню

```
export default function Sidebar({ isOpen, onClose }) {
  const { user, logout } = useAuth();
  const navigate = useNavigate();


  const menuItems = useMemo(() => {
    const base = [
      { path: '/home', icon: '🏠', label: 'Главная' },
      { path: '/profile', icon: '👤', label: 'Профиль' }
    ];


    if (user.role === 'student') {
      base.push(
        { path: '/courses', icon: '📚', label: 'Мои курсы' },
```

```
    { path: '/rating', icon: '🏆', label: 'Рейтинг' },
    { path: '/shop', icon: '🛒', label: 'Магазин' }
   );
  } else if (user.role === 'teacher') {
   base.push(
    { path: '/teacher-courses', icon: '📖', label: 'Преподавание' },
    { path: '/homework', icon: '📝', label: 'Проверка работ' }
   );
  } else if (user.role === 'admin') {
   base.push(
    { path: '/manage/courses', icon: '⚙️', label: 'Управление курсами' },
    { path: '/manage/students', icon: '👥', label: 'Студенты' },
    { path: '/manage/teachers', icon: '👨‍🏫', label: 'Преподаватели' }
   );
  }

  return base;
 }, [user.role]);

 return (
  <aside className={` sidebar ${isOpen ? 'open' : ''}` }>
   <nav>
    {menuItems.map(item => (
     <button
      key={item.path}
      onClick={() => {
```

```
            navigate(item.path);
            onClose();
          }}
        >
          <span>{item.icon}</span>
          <span>{item.label}</span>
        </button>
      ))}
    </nav>
    <button onClick={logout} className="logout-btn">
      🚪 Выйти
    </button>
  </aside>
 );
}
```

4.1.2 CourseCard — карточка курса

```
export default function CourseCard({ course, onClick, disabled = false }) {
 const ageCategory = Array.isArray(course.age_category)
  ? course.age_category.join(', ')
  : course.age_category;

 return (
  <div
   className={`course-card ${disabled ? 'disabled' : ''}`}
   onClick={disabled ? null : onClick}
```

```
 style={disabled ? { opacity: 0.6, cursor: 'not-allowed' } : {}}
>
 <CourseImage
  src={course.photo?.url}
  alt={course.name}
  className="course-card-image"
  placeholder="📚"
 />
 <div className="meta">
  <h3>{course.name}</h3>
  <p>{course.description?.substring(0, 60)}...</p>

  {!disabled && (
   <CourseProgressBar
    progress={course.progress || 0}
    lessonsTotal={course.lessons_count}
    lessonsCompleted={course.completed_lessons}
   />
  )}

  <div className="course-info">
   <span>👥 {ageCategory}</span>
   <span>📝 {course.lessons_count} уроков</span>
  </div>
 </div>
</div>
```

```
  );
}
```

4.1.3 CourseProgressBar — прогресс-бар

```
export default function CourseProgressBar({
  progress,
  lessonsTotal,
  lessonsCompleted
}) {
  const percentage = Math.min(100, Math.max(0, progress));

  return (
    <div className="course-progress">
      <div className="progress-info">
        <span>Прогресс: {percentage}%</span>
        <span>{lessonsCompleted} / {lessonsTotal} уроков</span>
      </div>
      <div className="progress-bar">
        <div
          className="progress-fill"
          style={{ width: `${percentage}%` }}
        />
      </div>
    </div>
  );
}
```

4.1.4 CourseImage — изображение с placeholder

```
export default function CourseImage({
  src,
  alt,
  placeholder = '📚',
  className = ''
}) {
  const [error, setError] = useState(false);
  const [loading, setLoading] = useState(true);

  if (error || !src) {
    return (
      <div className={`course-image-placeholder ${className}`}>
        <span className="placeholder-icon">{placeholder}</span>
      </div>
    );
  }

  return (
    <>
      {loading && (
        <div className={`course-image-placeholder ${className}`}>
          <span className="placeholder-icon">⏳</span>
        </div>
      )}
```

```
    <img
      src={src}
      alt={alt}
      className={` ${className} ${loading ? 'hidden' : ''}` }
      onLoad={() => setLoading(false)}
      onError={() => {
        setError(true);
        setLoading(false);
      }}
    />
  </>
 );
}
```

4.1.5 ConfirmModal — модальное окно подтверждения

```
export default function ConfirmModal({
 isOpen,
 onClose,
 onConfirm,
 title,
 message
}) {
 if (!isOpen) return null;

 return (
  <div className="modal-overlay" onClick={onClose}>
```

```
    <div className="modal-content" onClick={e => e.stopPropagation()}>
     <h2>{title}</h2>
     <p>{message}</p>
     <div className="modal-actions">
      <button onClick={onClose} className="btn-cancel">
       Отмена
      </button>
      <button onClick={onConfirm} className="btn-confirm">
       Подтвердить
      </button>
     </div>
    </div>
   </div>
  );
}
```

4.2 Компоненты-контейнеры (Smart Components)

SmartTopBar — "умная" верхняя панель

```
export default function SmartTopBar() {
  const { user } = useAuth();
  const [notifications, setNotifications] = useState([]);
  const [searchQuery, setSearchQuery] = useState('');
  const [showNotifications, setShowNotifications] = useState(false);

  useEffect(() => {
   loadNotifications();
```

```
  const interval = setInterval(loadNotifications, 30000);
  return () => clearInterval(interval);
}, []);

const loadNotifications = async () => {
  try {
    const data = await notificationService.getNotifications();
    setNotifications(data);
  } catch (error) {
    console.error('Error loading notifications:', error);
  }
};

const unreadCount = notifications.filter(n => !n.is_read).length;

return (
  <header className="smart-topbar">
    <SearchBox
      value={searchQuery}
      onChange={setSearchQuery}
      placeholder="Поиск курсов, преподавателей..."
    />

    <div className="topbar-actions">
      <NotificationBell
        count={unreadCount}
```

```
        onClick={() => setShowNotifications(!showNotifications)}
      />
      <UserMenu user={user} />
    </div>

    {showNotifications && (
      <NotificationDropdown
        notifications={notifications}
        onClose={() => setShowNotifications(false)}
      />
    )}
  </header>
 );
}
```

---

5. Context API для управления состоянием

5.1 AuthContext — контекст аутентификации

```
// src/contexts/AuthContext.js
import React, { createContext, useContext, useState, useEffect } from 'react';
import api from '../api/axiosInstance';

const AuthContext = createContext();

export const useAuth = () => useContext(AuthContext);

export function AuthProvider({ children }) {
```

```
const [user, setUser] = useState(undefined);
const [refreshing, setRefreshing] = useState(false);

// Проверка сессии при загрузке
useEffect(() => {
  api.get('/users/me')
    .then(({ data }) => setUser(data))
    .catch(() => setUser(null));
}, []);

// Автообновление токена каждые 4 минуты
useEffect(() => {
  if (user && !refreshing) {
    const id = setInterval(async () => {
      if (!refreshing) {
        setRefreshing(true);
        try {
          const { data } = await api.post('/users/refresh');
          setUser(data);
        } catch (err) {
          console.error('Token refresh failed:', err);
          if (err.response?.status === 401) {
            setUser(null);
          }
        } finally {
          setRefreshing(false);
```

```
      }
    }
  }, 4 * 60 * 1000);
  return () => clearInterval(id);
 }
}, [user, refreshing]);

const login = async (username, password) => {
 const { data } = await api.post('/users/auth', {
  username,
  password
 });
 setUser(data);
 return data;
};

const logout = async () => {
 try {
  await api.post('/users/logout');
 } catch (err) {
  console.error('Logout error:', err);
 } finally {
  setUser(null);
 }
};
```

```
  const updateUser = async () => {
    try {
      const { data } = await api.get('/users/me');
      setUser(data);
      return data;
    } catch (err) {
      console.error('Update user error:', err);
      throw err;
    }
  };

  return (
    <AuthContext.Provider value={{ user, login, logout, updateUser }}>
      {children}
    </AuthContext.Provider>
  );
}
```

Использование в компонентах:

```
function SomeComponent() {
  const { user, logout } = useAuth();

  if (!user) return <div>Loading...</div>;

  return (
    <div>
      <p>Привет, {user.first_name}!</p>
```

```
    <button onClick={logout}>Выйти</button>
   </div>
  );
}
```

5.2 NotificationsContext — контекст уведомлений

```
// src/contexts/NotificationsContext.js
import React, { createContext, useContext, useState, useEffect } from 'react';
import notificationService from '../services/notificationService';

const NotificationsContext = createContext();

export const useNotifications = () => useContext(NotificationsContext);

export function NotificationsProvider({ children }) {
  const [notifications, setNotifications] = useState([]);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    loadNotifications();
    const interval = setInterval(loadNotifications, 30000);
    return () => clearInterval(interval);
  }, []);

  const loadNotifications = async () => {
    try {
```

```
    const data = await notificationService.getNotifications();
    setNotifications(data);
  } catch (error) {
    console.error('Error loading notifications:', error);
  } finally {
    setLoading(false);
  }
};

const markAsRead = async (id) => {
  try {
    await notificationService.markAsRead(id);
    setNotifications(prev =>
      prev.map(n => n.id === id ? { ...n, is_read: true } : n)
    );
  } catch (error) {
    console.error('Error marking as read:', error);
  }
};

const markAllAsRead = async () => {
  try {
    await notificationService.markAllAsRead();
    setNotifications(prev =>
      prev.map(n => ({ ...n, is_read: true }))
    );
```

```
    } catch (error) {
      console.error('Error marking all as read:', error);
    }
  };

  const unreadCount = notifications.filter(n => !n.is_read).length;

  return (
    <NotificationsContext.Provider value={{
      notifications,
      loading,
      unreadCount,
      markAsRead,
      markAllAsRead,
      refresh: loadNotifications
    }}>
      {children}
    </NotificationsContext.Provider>
  );
}
```

---

6. Custom Hooks (Composables)

6.1 useConfirm — подтверждение действий

```
// src/hooks/useConfirm.js
import { useState, useCallback } from 'react';
```

```
export function useConfirm() {
  const [isOpen, setIsOpen] = useState(false);
  const [config, setConfig] = useState({
    title: '',
    message: '',
    onConfirm: null
  });

  const confirm = useCallback(({ title, message, onConfirm }) => {
    setConfig({ title, message, onConfirm });
    setIsOpen(true);

    return new Promise((resolve) => {
      setConfig(prev => ({
        ...prev,
        onConfirm: () => {
          onConfirm?.();
          setIsOpen(false);
          resolve(true);
        }
      }));
    });
  }, []);

  const cancel = useCallback(() => {
    setIsOpen(false);
```

```
 }, []);

 return {
  isOpen,
  config,
  confirm,
  cancel
 };
}
```

Использование:

```
function MyComponent() {
 const { isOpen, config, confirm, cancel } = useConfirm();

 const handleDelete = async () => {
  await confirm({
   title: 'Удаление курса',
   message: 'Вы уверены, что хотите удалить этот курс?',
   onConfirm: async () => {
    await courseService.deleteCourse(courseId);
    toast.success('Курс удалён');
   }
  });
 };

 return (
  <>
```

```
    <button onClick={handleDelete}>Удалить</button>
    <ConfirmModal
     isOpen={isOpen}
     onClose={cancel}
     onConfirm={config.onConfirm}
     title={config.title}
     message={config.message}
    />
   </>
  );
}
```

6.2 useMobileKeyboard — обработка виртуальной клавиатуры

```
// src/hooks/useMobileKeyboard.js
import { useEffect, useState } from 'react';

export function useMobileKeyboard() {
  const [isKeyboardVisible, setIsKeyboardVisible] = useState(false);

  useEffect(() => {
    const handleResize = () => {
      // На мобильных устройствах высота окна меняется при появлении клавиатуры
      const viewportHeight = window.visualViewport?.height || window.innerHeight;
      const windowHeight = window.innerHeight;

      setIsKeyboardVisible(viewportHeight < windowHeight * 0.8);
```

```
    };

    window.visualViewport?.addEventListener('resize', handleResize);
    window.addEventListener('resize', handleResize);

    return () => {
      window.visualViewport?.removeEventListener('resize', handleResize);
      window.removeEventListener('resize', handleResize);
    };
  }, []);

  return isKeyboardVisible;
}
```

Использование:

```
function SearchComponent() {
  const isKeyboardVisible = useMobileKeyboard();

  return (
    <div className={` search-container ${isKeyboardVisible ? 'keyboard-open' : ''}` }>
      <input type="text" placeholder="Поиск..." />
    </div>
  );
}
```

6.3 Дополнительные custom hooks (примеры)

```
// useDebounce - отложенное выполнение
```

```
export function useDebounce(value, delay = 500) {
  const [debouncedValue, setDebouncedValue] = useState(value);

  useEffect(() => {
    const timer = setTimeout(() => {
      setDebouncedValue(value);
    }, delay);

    return () => clearTimeout(timer);
  }, [value, delay]);

  return debouncedValue;
}

// useLocalStorage - работа с localStorage
export function useLocalStorage(key, initialValue) {
  const [value, setValue] = useState(() => {
    try {
      const item = window.localStorage.getItem(key);
      return item ? JSON.parse(item) : initialValue;
    } catch {
      return initialValue;
    }
  });

  const setStoredValue = (newValue) => {
```

```
    try {
      setValue(newValue);
      window.localStorage.setItem(key, JSON.stringify(newValue));
    } catch (error) {
      console.error('Error saving to localStorage:', error);
    }
  };

  return [value, setStoredValue];
}

// useIntersectionObserver - отслеживание видимости элемента
export function useIntersectionObserver(ref, options = {}) {
  const [isVisible, setIsVisible] = useState(false);

  useEffect(() => {
    if (!ref.current) return;

    const observer = new IntersectionObserver(([entry]) => {
      setIsVisible(entry.isIntersecting);
    }, options);

    observer.observe(ref.current);

    return () => observer.disconnect();
  }, [ref, options]);
```

```
  return isVisible;
}
```

7. Интеграция с API (Axios)

7.1 Service layer

Все API запросы вынесены в отдельные сервисы:

```
// src/services/courseService.js
import api from '../api/axiosInstance';

export async function listStudentCourses() {
  const { data } = await api.get('/courses/student');
  return data;
}

export async function getCourseDetails(courseId) {
  const { data } = await api.get(` /courses/${courseId}` );
  return data;
}

export async function getStudentLessonProgress() {
  const { data } = await api.get('/lesson-groups/student/progress');
  return data;
}

export async function getAllCoursesFiltered(user, limit = 100, offset = 0) {
```

```
  const { data } = await api.get('/courses/', {
    params: { limit, offset }
  });
  return data;
}

export default {
  listStudentCourses,
  getCourseDetails,
  getStudentLessonProgress,
  getAllCoursesFiltered
};
```

7.2 Использование в компонентах

```
import { listStudentCourses, getStudentLessonProgress } from '../services/courseService';

export default function StudentCoursesPage() {
  const [courses, setCourses] = useState([]);
  const [progress, setProgress] = useState([]);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    (async () => {
      try {
        setLoading(true);
        const [coursesData, progressData] = await Promise.all([
```

```
        listStudentCourses(),
        getStudentLessonProgress()
      ]);
      setCourses(coursesData);
      setProgress(progressData);
    } catch (error) {
      console.error('Error loading data:', error);
    } finally {
      setLoading(false);
    }
  })();
 }, []);

 // Render...
}
```

8. Оптимизация производительности

8.1 useMemo для вычисляемых значений

```
const filteredCourses = useMemo(() => {
 return courses.filter(course => {
  const matchesSearch = course.name
    .toLowerCase()
    .includes(searchQuery.toLowerCase());

  const matchesFilter = selectedFilter === 'all' ||
    course.category === selectedFilter;
```

```
    return matchesSearch && matchesFilter;
  });
}, [courses, searchQuery, selectedFilter]);
```

8.2 useCallback для стабильных функций

```
const handleCourseClick = useCallback((courseId) => {
  navigate(`/courses/${courseId}/student`);
}, [navigate]);

const handleSearch = useCallback((query) => {
  setSearchQuery(query);
}, []);
```

8.3 React.memo для предотвращения рендеров

```
const CourseCard = React.memo(({ course, onClick }) => {
  return (
    <div className="course-card" onClick={() => onClick(course.id)}>
      <h3>{course.name}</h3>
      <p>{course.description}</p>
    </div>
  );
});
```

8.4 Lazy loading компонентов

```
import React, { lazy, Suspense } from 'react';
```

```
const HeavyComponent = lazy(() => import('./HeavyComponent'));

export default function Page() {
  return (
    <Suspense fallback={<div>Загрузка...</div>}>
      <HeavyComponent />
    </Suspense>
  );
}
```

9. Формы и валидация

9.1 Управляемые формы

```
export default function ProfileEditForm({ initialData, onSave }) {
  const [formData, setFormData] = useState(initialData);
  const [errors, setErrors] = useState({});

  const handleChange = (field, value) => {
    setFormData(prev => ({ ...prev, [field]: value }));
    // Очищаем ошибку при изменении
    if (errors[field]) {
      setErrors(prev => ({ ...prev, [field]: null }));
    }
  };

  const validate = () => {
```

```
  const newErrors = {};

  if (!formData.first_name) {
    newErrors.first_name = 'Имя обязательно';
  }

  if (!formData.email) {
    newErrors.email = 'Email обязателен';
  } else if (!/\S+@\S+\.\S+/.test(formData.email)) {
    newErrors.email = 'Некорректный email';
  }

  setErrors(newErrors);
  return Object.keys(newErrors).length === 0;
};

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

  if (!validate()) return;

  try {
    await onSave(formData);
    toast.success('Профиль обновлён');
  } catch (error) {
    toast.error('Ошибка при сохранении');
```

```
  }
};

return (
  <form onSubmit={handleSubmit}>
    <div className="form-group">
      <label>Имя</label>
      <input
        type="text"
        value={formData.first_name}
        onChange={e => handleChange('first_name', e.target.value)}
      />
      {errors.first_name && (
        <span className="error">{errors.first_name}</span>
      )}
    </div>

    <div className="form-group">
      <label>Email</label>
      <input
        type="email"
        value={formData.email}
        onChange={e => handleChange('email', e.target.value)}
      />
      {errors.email && (
        <span className="error">{errors.email}</span>
```

```
      )}
    </div>

    <button type="submit">Сохранить</button>
   </form>
  );
}
```

10. Примеры реализованных страниц

10.1 HomePage — главная страница

```
export default function HomePage() {
  const { user } = useAuth();
  const [events, setEvents] = useState([]);
  const [news, setNews] = useState([]);
  const [studentData, setStudentData] = useState(null);

  // Множественная загрузка данных
  useEffect(() => {
   if (!user) return;

   const loadData = async () => {
    try {
     const [scheduleData, newsData, studentInfo] = await Promise.all([
       getUserScheduleOptimized(user),
       api.get('/news/').then(res => res.data),
       user.role === 'student'
```

```
        ? api.get('/students/me').then(res => res.data)
        : Promise.resolve(null)
      ]);

      setEvents(scheduleData);
      setNews(newsData);
      setStudentData(studentInfo);
    } catch (error) {
      console.error('Error loading homepage data:', error);
    }
  };

  loadData();
}, [user]);

return (
  <div className="page-wrapper">
    <Sidebar />
    <main className="main-content">
      <SmartTopBar />

      <div className="home-grid">
        <section className="schedule-section">
          <h2>Расписание</h2>
          <Schedule events={events} />
        </section>
```

```
        <aside className="sidebar-widgets">
         {user.role === 'student' && (
           <BestCoins points={studentData?.points || 0} />
         )}

         <div className="news-widget">
           <h3>Новости</h3>
           {news.slice(0, 5).map(item => (
             <NewsCard key={item.id} news={item} />
           ))}
         </div>
        </aside>
       </div>
     </main>
   </div>
  );
}
```

10.2 StudentCoursesPage — курсы студента

```
export default function StudentCoursesPage() {
  const { user } = useAuth();
  const navigate = useNavigate();
  const [myCourses, setMyCourses] = useState([]);
  const [otherCourses, setOtherCourses] = useState([]);
  const [loading, setLoading] = useState(true);
```

```
useEffect(() => {
  (async () => {
    try {
      setLoading(true);

      const [available, progress, all] = await Promise.all([
        listStudentCourses(),
        getStudentLessonProgress(),
        getAllCoursesFiltered(user, 100, 0)
      ]);

      const availableIds = new Set(available.map(c => c.id));

      setMyCourses(available);
      setOtherCourses(
        all.objects.filter(c => !availableIds.has(c.id))
      );
    } catch (error) {
      console.error('Error loading courses:', error);
    } finally {
      setLoading(false);
    }
  })();
}, [user]);
```

```
if (loading) return <LoadingSpinner />;

return (
 <div className="page-wrapper">
  <Sidebar />
  <main className="main-content">
   <SmartTopBar />

   <section className="courses-section">
    <h2>Мои курсы</h2>
    <div className="courses-grid">
     {myCourses.map(course => (
      <CourseCard
       key={course.id}
       course={course}
       onClick={() => navigate(`/courses/${course.id}/student`)}
      />
     ))}
    </div>
   </section>

   <section className="courses-section">
    <h2>Доступные курсы</h2>
    <div className="courses-grid">
     {otherCourses.map(course => (
      <CourseCard
```

```
            key={course.id}
            course={course}
            disabled
            onClick={() => alert('Обратитесь к администратору')}
          />
        ))}
      </div>
    </section>
  </main>
 </div>
 );
}
```

11. Конфигурация и зависимости

11.1 package.json

```
{
 "name": "front_it_school",
 "version": "0.1.0",
 "private": true,
 "dependencies": {
  "@fullcalendar/daygrid": "^6.1.17",
  "@fullcalendar/interaction": "^6.1.17",
  "@fullcalendar/list": "^6.1.19",
  "@fullcalendar/react": "^6.1.17",
  "@fullcalendar/timegrid": "^6.1.17",
  "axios": "^1.9.0",
```

```
    "react": "^19.1.0",
    "react-dom": "^19.1.0",
    "react-router-dom": "^7.6.0",
    "react-toastify": "^11.0.5"
  },
  "scripts": {
    "start": "craco start",
    "build": "craco build",
    "test": "craco test"
  }
}
```

11.2 craco.config.js

```
const CssMinimizerPlugin = require('css-minimizer-webpack-plugin');


module.exports = {
  webpack: {
    configure: (webpackConfig) => {
      webpackConfig.optimization.minimizer.push(
        new CssMinimizerPlugin()
      );
      return webpackConfig;
    }
  }
};
```

---

12. Выводы

В ходе выполнения третьей лабораторной работы были освоены следующие технологии и подходы.