



**Министерство науки и высшего образования Российской
Федерации Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Московский государственный технический
университет имени Н.Э. Баумана
(национальный исследовательский
университет)» (МГТУ им. Н.Э. Баумана)**

ФАКУЛЬТЕТ _____ Информатика и системы управления

КАФЕДРА _____ Системы обработки информации и управления

Лабораторная работа №8

По курсу «Разработка интернет приложений»

Подготовила:

Студентка группы ИУ5-55Б.

Очеретная С.В.

18.12.2020

Проверил:

Преподаватель кафедры ИУ5

Гапанюк Ю.Е.

Москва, 2021 г.

Цель лабораторной работы: изучение возможностей создания пользовательского интерфейса в веб-приложениях с использованием библиотеки React.

Задание:

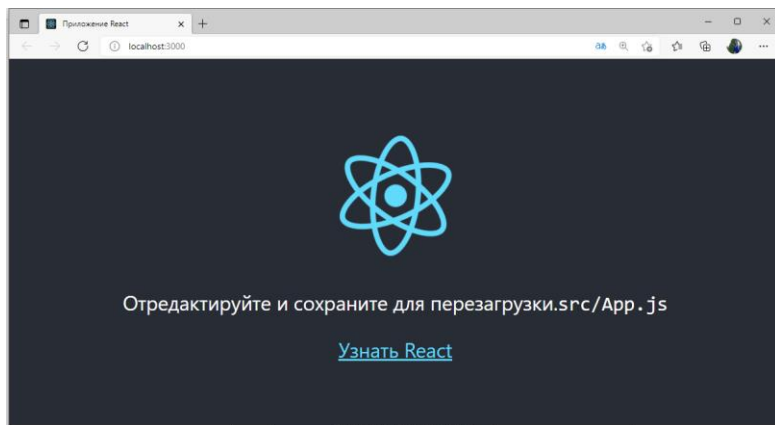
Разработать React-приложение. Для создания приложения необходимо решить следующие задачи:

1. Создать стартовый React-проект. Удалить неиспользуемый код. Организовать директории для страниц, компонентов, утилит и работы с сетью.
2. Организовать роутинг в веб-приложении.
3. Разработать базовые страницы, на которых будут отображаться сущности из выбранной вами предметной области:
 - i. Стартовая страница.
 - ii. Страница просмотра списка объектов.
 - iii. Страница просмотра конкретного объекта.
4. Вынести переиспользуемые компоненты в отдельные файлы:
 - i. Для навигации по приложению можно добавить header.
 - ii. Для отображения дополнительной информации (данные о студенте и предметной области) можно использовать footer.
 - iii. Источники ввода-вывода (поля ввода (inputs)/формы/текстовые блоки).
 - iv. Переиспользуемые таблицы/гриды.
5. Добавить асинхронные запросы в разработанный API, чтобы страница получала данные с сервера.
6. Если в Вашем проекте реализована сложная логика работы с состоянием приложения, то рекомендуется добавить пользовательские хуки.
7. Страницы приложения должны хорошо отображаться как на больших, так и на маленьких экранах.

Ход работы

1) Создание React-приложения

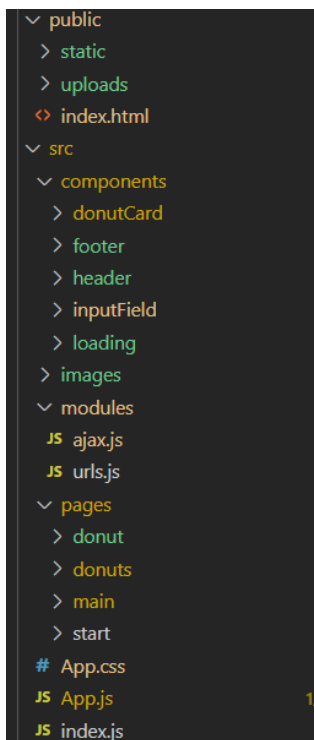
```
npx create-react-app lab8  
cd lab8  
npm start
```



Создать оптимизированную сборку приложения:
`npm run build`

Запустить локальный сервер для разработки:
`npm run start`

Почистим ненужные файлы и организуем структуру папок. Получим следующую структуру приложения:



2) Роутинг в react приложении

Установим библиотеки react-router-dom и react-router
`npm install react-router react-router-dom`

Сделаем так, что App.js будет отвечать за роутинг в нашем приложении

```

return (
  <BrowserRouter basename="/" >
    <Header/>
    <Switch>
      <Route exact path="/">
        {loading && <Loading/>}
        {!loading && <MainPage/>}
      </Route>
      <Route exact path="/donuts">
        {loading && <Loading/>}
        {!loading && <DonutsPage donuts={donuts}/>}
      </Route>
      <Route path="/donuts/:donutId">
        {loading && <Loading/>}
        {!loading && <DonutPage donuts={donuts}/>}
      </Route>
    </Switch>
    <Footer/>
  </BrowserRouter>
);
}

```

Роутер позволяет нам перемещаться между разными страницами приложения без перезагрузки. Можем переходить по ним без перезагрузки, используя Link

```

<li className = "nav-item me-2">
  <Link to="/">
    <div className="btn my-btn">
      Главная страница
    </div>
  </Link>
</li>
<li className = "nav-item me-2">
  <Link to="/donuts">
    <div className="btn my-btn">
      Все пончики
    </div>
  </Link>
</li>
<li className = "nav-item me-2">
  <Link to="/donuts/1">
    <div className="btn my-btn">
      1 пончик
    </div>
  </Link>

```

3) Базовые страницы приложения

- Создадим директорию со страницами нашего приложения src/pages
- Создадим стартовую страницу с поиском пончиков

Donuts World


Главная страница

Все пончики

1 пончик


Log out

Settings

Светлана

Welcome to Donuts World!

Искать



Пончик Шоколадка

130 руб.

Подробнее

Developer: Ocheretnaya Svetlana; Предметная область: Пончики :)

Код страницы:

```

function MainPage() {
  const [searchValue, setSearchValue] = useState('Шоколадка');
  const [loading, setLoading] = useState(false);
  const [donuts, setDonuts] = useState([]);

  const handleSearch = () => {
    setLoading(true);
    ajax.get({url: urls.donuts()}).then(({response}) => {
      console.log("response:")
      console.log(response)
      setDonuts(response.filter(item => item.name && item.name.toLowerCase().includes(searchValue.toLowerCase())));
      setLoading(false)
    });
  };

  return (
    <div className='my-background d-flex justify-content-center flex-column align-items-center'>
      <h1 className='welcome-title'>Welcome to Donuts World!</h1>
      <InputField value={searchValue} setValue={setSearchValue} placeholder="поиск" loading={loading} onSubmit={handleSearch} buttonTitle="Искать" />
      {loading && <Loading/>}
      {!loading && !donuts.length ? <h1 className='welcome-error'>Пончики не найдены</h1>:
      <div className="d-flex justify-content-center flex-wrap">
        {donuts?.map((item, index)=>{
          return(
            <div key={index}>
              <DonutCard donutLoad={item}/>
            </div>
          )
        })}
      </div>
      }
    </div>
  );
}

export default MainPage;

```

- Добавим переход по странице на страницу 1го пончика и в хедере для примера

```

<Link to="/">
  <div className="btn my-btn">
    Главная страница
  </div>
</Link>

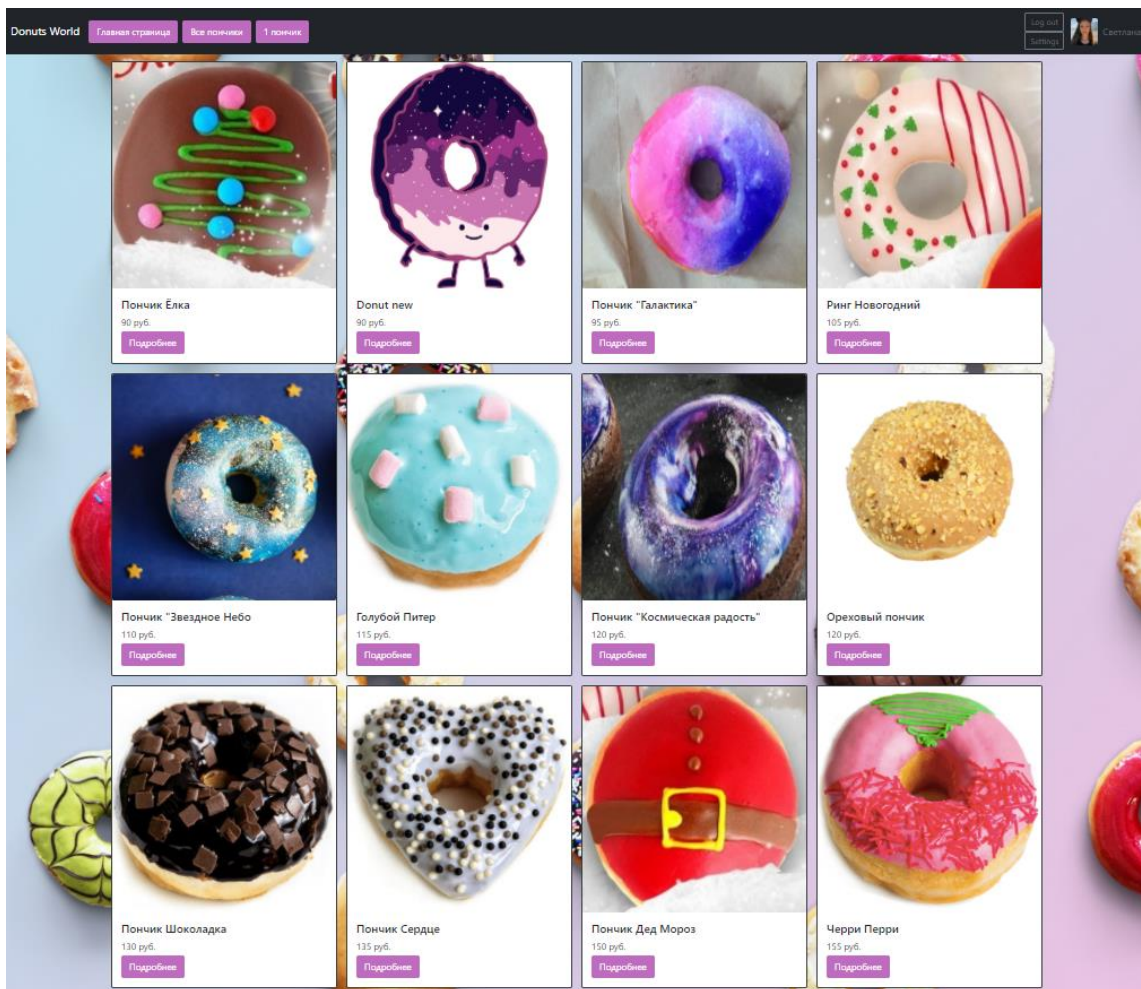
```

```

<Link to="/">
  <div className="btn my-btn" href="/">Перейти на главную страницу</div>
</Link>
</div>

```

- Создадим страницу просмотра всех пончиков



Код страницы:

```
import DonutCard from '../components/donutCard/donutCard';

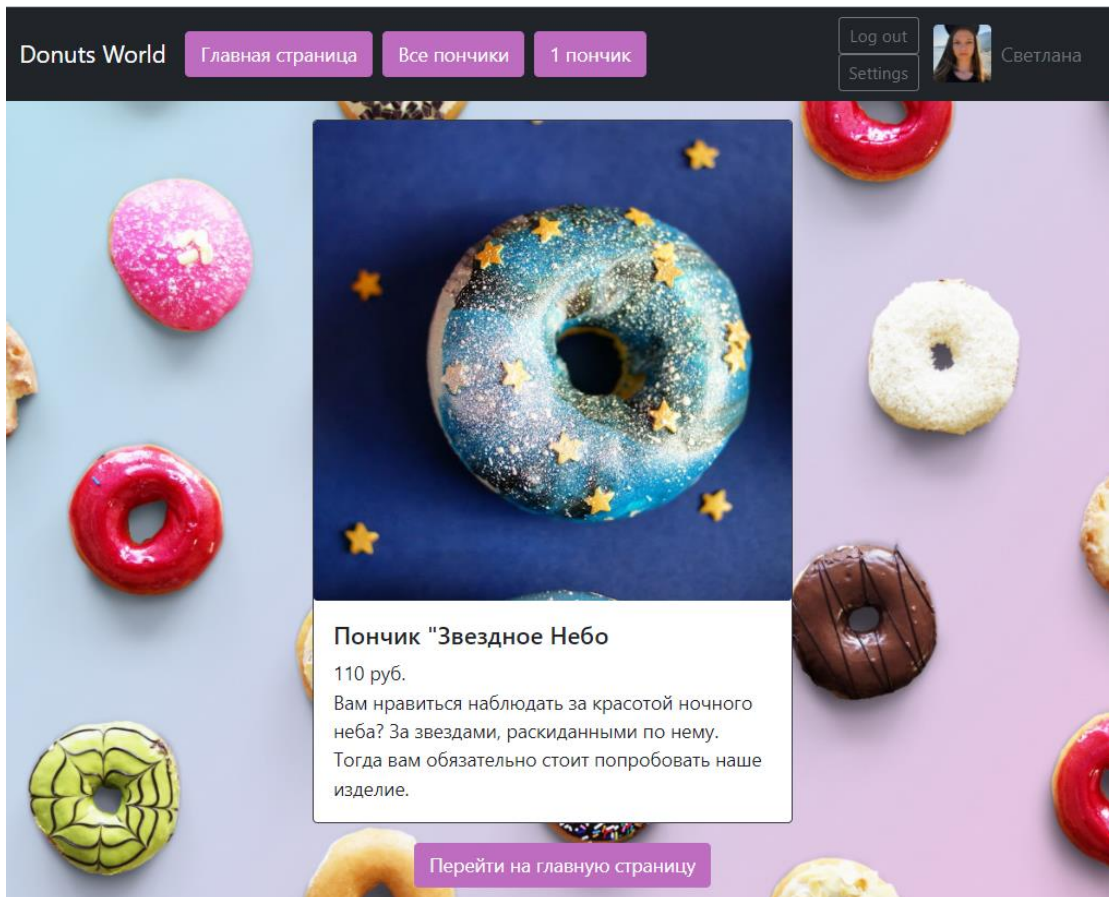
const DonutsPage = ({donuts}) => {
  return (
    <div className='my-background'>
      {!donuts.length ? <h1>Пончики не найдены</h1>:
        <div className="d-flex justify-content-center flex-wrap mt-3">
          {donuts?.map((item, index)=>{
            return(
              <div key={index}>
                <DonutCard donutLoad={item}/>
              </div>
            )
          })}
        </div>
      }
    </div>
  );
};

export default DonutsPage;
```

- Переход по странице в хедере


```
<Link to="/donuts">
  <div className="btn my-btn">
    Все пончики
  </div>
</Link>
```

- Создадим страницу просмотра конкретного пончика



- Переход по странице

```
{!isDonutPage &&
<Link to={` /donuts/${donut.pk}`} >
  <div className="btn my-btn mt-1">Подробнее</div>
</Link>
}
```

Код страницы:


```

import { useParams } from "react-router-dom";
import { Link } from "react-router-dom";
import DonutCard from '../components/donutCard/donutCard';

const DonutPage = ({ donuts }) => {
  const { donutId } = useParams();
  console.log("donut ID:");
  console.log({ donutId }, typeof donutId);
  const donut = donuts.find((d) => d.pk + '' === donutId)

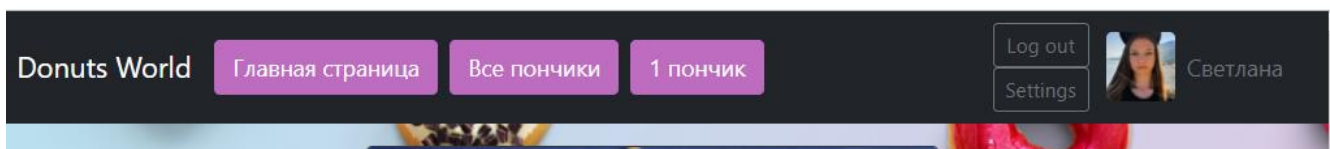
  return (
    <div className='my-background'>
      <div className="d-flex justify-content-center mt-3 flex-column align-items-center">
        <DonutCard donutLoad={donut} isDonutPage={true}/>
        <Link to="/">
          <div className="btn my-btn" href="/">Перейти на главную страницу</div>
        </Link>
      </div>
    </div>
  );
}

export default DonutPage;

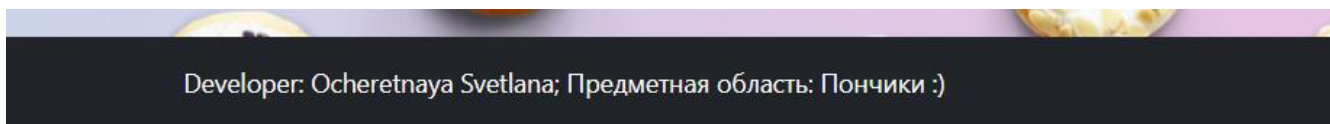
```

4. Вынесли все переиспользуемые компоненты в отдельные файлы:

- Для навигации по приложению добавили header



- Для отображения дополнительной информации (данные о студенте и предметной области) добавили footer



- Вынесли inputField в отдельный компонент

```

import { Button } from "react-bootstrap";
import React from "react";

const InputField = ({ value, setValue, onSubmit, loading, placeholder, buttonText = 'Поиск' }) => {
  return <div className="input-field mb-3 d-flex">
    <input className="form-control" value={value} placeholder={placeholder} onChange={(event) => setValue(event.target.value)}/>
    <div className="input-group-append">
      <Button className="my-btn" disabled={loading} onClick={onSubmit}>{buttonText}</Button>
    </div>
  </div>
}

export default InputField

```

- Вынесли колесико загрузки в отдельный компонент

```
import {Spinner} from 'react-bootstrap'

const Loading = () => {
  return <div className="loadingBg"><Spinner animation="border"/></div>
}

export default Loading
```

- Вынесли карточку пончика в отдельный компонент

```
const DonutCard = ({donutLoad, isDonutPage=false}) => {
  const [donut, setDonut] = useState({
    pk: 1,
    name: "Donut name",
    cost: 0,
    info: 'About donut',
    picture: '/static/img/donut-default.jpg'
  });

  useEffect(() => {
    if (donutLoad) {
      setDonut(donutLoad);
    }
  }, [donutLoad]);

  return <Card className="card-donut">
    <Card.Img src = {donut.picture} className="card-img donut-img" alt={donut.name} height={100} width={100}/>
    <Card.Body>
      <div className="textStyle">
        <Card.Title className="card-title">{donut.name}</Card.Title>
      </div>
      <div className="textStyle">
        <Card.Text>{donut.cost} py6.</Card.Text>
      </div>
      {isDonutPage &&
      <div className="textStyle">
        <Card.Text>{donut.info}</Card.Text>
      </div>
      }
      {!isDonutPage &&
      <Link to={` /donuts/${donut.pk}`} >
        <div className="btn my-btn mt-1">Подробнее</div>
      </Link>
      }
    </Card.Body>
  </Card>
}

export default DonutCard;
```

5. Добавим асинхронные запросы в разработанный API, чтобы страница получала данные с сервера.

Загрузка пончиков производится на главной странице

```
import ajax from "../modules/ajax";
import urls from "../modules/urls";

function App() {
  const [loading, setLoading] = useState(false);
  const [donuts, setDonuts] = useState([]);

  const donutsLoading = () => {
    setLoading(true);
    ajax.get({url: urls.donuts()}).then(({response}) => {
      setDonuts(response)
      setLoading(false)
    });
  }

  useEffect(() => {
    donutsLoading();
  }, []);
}
```

А также при поиске на главной странице:

```
import ajax from '../modules/ajax';
import urls from '../modules/urls';

function MainPage() {
  const [searchValue, setSearchValue] = useState('Шоколадка');
  const [loading, setLoading] = useState(false);
  const [donuts, setDonuts] = useState([]);

  const handleSearch = () => {
    setLoading(true);
    ajax.get({url: urls.donuts()}).then(({response}) => {
      console.log("response:")
      console.log(response)
      setDonuts(response.filter(item => item.name && item.name.toLowerCase().includes(searchValue.toLowerCase())));
      setLoading(false)
    });
  }

  return (
    <div className='my-background d-flex justify-content-center flex-column align-items-center'>
      <h1 className='welcome-title'>Welcome to Donuts World!</h1>
      <TextField value={searchValue} setValue={setSearchValue} placeholder="поиск" loading={loading} onSubmit={handleSearch} buttonTitle="Искать" />
    </div>
  )
}
```

Производится fetch-запрос аналогично с 7 лабораторной в файлике module/ajax.js

- Приложение отображается корректно на любых устройствах. Для этого использовалась bootstrap-верстка и media-запросы в файлике src/App.css

