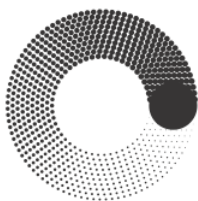


ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ



МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

*Факультет Информационных технологий  
Кафедра Информатики и информационных технологий*

направление подготовки

09.03.02 «Информационные системы и технологии»

## КУРСОВОЙ ПРОЕКТ

Дисциплина: \_\_\_\_\_ Базы \_\_\_\_\_ данных \_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

Тема: \_\_\_\_\_ Создание базы данных для хранения информации о товарах  
для веб-приложения \_\_\_\_\_

\_\_\_\_\_

Выполнили: студенты группы \_\_221-3710\_\_

\_\_\_\_ Науменко.Н \_\_\_\_\_  
(Фамилия И.О.)

\_\_\_\_ Гавриченко.А \_\_\_\_\_  
(Фамилия И.О.)

Проверил: \_\_\_\_\_  
(Фамилия И.О., степень, звание)

Дата, подпись \_\_\_\_\_  
(Дата) (Подпись)

Замечания: \_\_\_\_\_

Москва

2024

## Оглавление

Введение.....	3
Теоретическая часть: .....	4
<b>1.1. Анализ рынка.</b> .....	4
<b>1.2 Выбор инструментов разработки.</b> .....	6
Практическая часть: .....	7
Заключение: .....	14
Памятка:.....	14
<b>BACKEND.</b> .....	14
Источники: .....	27

## Введение

Тема Проекта – создание базы данных для хранения информации о товарах для веб-приложения

В наше время использование баз данных обрело значительное удобство и доступность для каждого. Притом, они предлагают безграничные возможности для для таких сфер, как, к примеру бизнес товаров, так как можно без особых затрат хранить огромные массивы информации и удобно получать к ней доступ. При помощи базы данных можно оптимизировать многие процессы, начиная от складских , заканчивая процессами покупки товара.

Допустим что у нас есть информация о товаре. Как удобно отображать ее для покупателя? Здесь нам поможет СУБД, которое упростит работу с базой данных и отправкой информации на сайт.

Цель проекта – создать fullstack веб-приложение для отображения данных о товарах из СУБД

Исходя из цели формируются следующие задачи:

- 1.Изучение предметной области.
2. Выбор и обоснование инструментов разработки.
3. Проектирование проекта.
4. Реализация физической модели данных.

## Теоретическая часть:

### 1.1. Анализ рынка.

Доля онлайн-продаж в ритейле продолжает расти: в 2022 году на них пришлось 15% от всего ритейла и 30% от рынка непродовольственного ритейла (+3 п.п. и +5 п.п. год к году соответственно). Все данные указаны без учета продаж автомобилей и топлива.

Объем российской интернет-торговли в 2023 году составил 7,4 трлн рублей, увеличившись на 30% в сравнении с предыдущим годом. Такую оценку рынку в компании IBC Real Estate дали в середине февраля 2024 года. По оценкам экспертов, доля универсальных маркетплейсов (Wildberries, Ozon, «Яндекс Маркет», Aliexpress Russia) в общем объеме онлайн-покупок в РФ по итогам 2023 года достигла 58%.

Объем российского рынка электронной коммерции по итогам 2023 года достиг 6,36 трлн рублей, что на 28% больше в сравнении с 2022-м. Об этом свидетельствуют данные Ассоциации компаний интернет-торговли (АКИТ), которые были обнародованы в феврале 2024 года. Исследование подготовлено при участии «Сбера».

В апреле 2021 года исследовательское агентство Data Insight обнародовало обновленный рейтинг крупнейших интернет-магазинов в России. Первое место по итогам 2020 года сохранила компания Wildberries с выручкой в 413,2 млрд рублей, что на 96% больше показателя годичной давности.

Ozon опередил «Ситилинк» и занял второе место в списке, зарегистрировав выручку в 197 млрд рублей. В сравнении с 2019 годом она увеличились на 144%. У «Ситилинка» доходы поднялись на 47%, до 132,7 млрд рублей.

Местами поменялись «М.Видео» и DNS: в 2020 году DNS нарастил онлайн-продажи на 117%, до 116,7 млрд рублей (12 млн заказов), а у «М.Видео» рост был двукратным, до 113,2 млрд рублей (8,9 млн заказов). Интернет-магазин «Эльдорадо» поднялся с 8-го на 6-е место. Согласно докладу, выдержки из которого приводит «Интерфакс», на онлайн-сегмент пришлось 13,8% розничных продаж в РФ по итогам 2023 года против 11,6% годом ранее. Доля онлайн-продаж внутри России в 2023 году составила 96,9%, а трансграничная торговля заняла 3,1% отечественного рынка электронной коммерции.

Российский рынок интернет-торговли (млрд руб.)

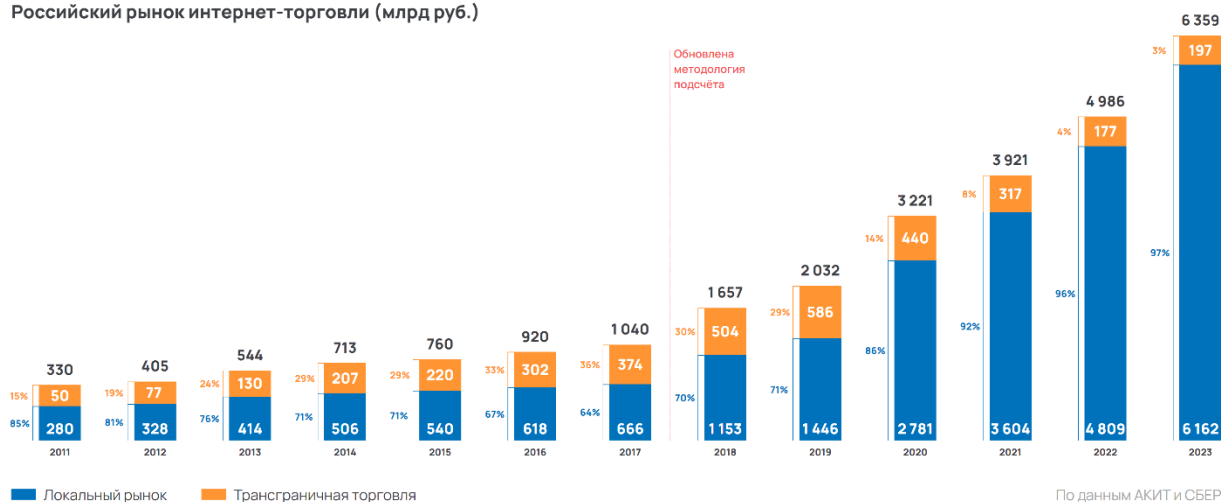


Рис.1.

### Оценка российского рынка интернет-торговли.

В исследовании отмечается, что рост популярности в России и изменение региональной структуры продаж стали заметны, начиная с 2020 года: все это время компании интернет-торговли активно инвестируют в развитие своей логистической инфраструктуры на местах, развивают партнерства с локальными бизнесами, — это напрямую влияет на повышение качества сервисов и расширение ассортиментных предложений для всех россиян.

Определенной областью можно не ограничиваться, ведь такое веб-приложение можно создать как для товаров электроники или одежды, так и для стройматериалов или промышленной техники. В нашем случае будет

реализован сайт с продажей электронных услуг, таких как прокачка уровней в видеоиграх или покупка игровой валюты.

У каждого товара будет изображение, цена и название. В базе данных помимо этого будет храниться номер товара. Изображения не будут храниться на сайте, а подгружаться из интернета по ссылке, что облегчает работу с ними. Для цены и названия есть лимиты в \* символов для облегчения веса файла базы данных.

Имена и учетные данные клиента не потребуются, так как это все будет проходить напрямую к работнику и все детали будут обговариваться лично, сайт является лишь агрегатором в данном случае.

### 1.2 Выбор инструментов разработки.

Для реализации этого проекта был выбран один из самых популярных и стабильных СУБД – MySQL. Он обладает хорошей производительностью и быстродействием благодаря оптимизированным алгоритмам выполнения запросов. Также, MySQL умеет работать с большими объемами данных с минимальными задержками. в MySQL реализованы различные типы данных, индексы, хранимые процедуры, триггеры и другие функции, обеспечивающие гибкость при обработке данных. Это дает возможность создания сложных схем данных. MySQL совместима с довольно большим количеством операционных систем. Рассматриваемый продукт имеет открытый исходный код и может быть использован бесплатно. Это делает его доступным для большого круга разработчиков и пользователей.

Для разработки frontend части приложения был выбран vite + react. React — это JavaScript-библиотека для создания пользовательских интерфейсов. Язык, используемый в react, один из самых доступных и простых для понимания.

React.js работает на идее декларативного подхода. Это называется «UI — функция данных» — разработчик описывает, как ведёт себя интерфейс в зависимости от данных и событий. React гибкий, потому что позволяет выбирать, какие библиотеки использовать и легко менять инструменты разработки, удобный, потому что за 11 лет в react есть решения практически на все идеи, имеется огромное количество инструментов, Читаемый, так как код

имеет блочную структуру и намного чаще ошибка ведет к нужному неисправному элементу.

Для backend части приложения был взят Nodejs из-за высокой совместимости с frontend частью. Мало того весь проект будет написан на одном языке – JavaScript, что облегчает разработку приложения

## Практическая часть:

Для начала надо настроить локальный сервер для реализации базы данных. Для этого был выбран самый удобный вариант: приложение XAMPP, где без труда можно запустить Apache+MySQL сервер, и настраивать базу данных через phpMyAdmin.

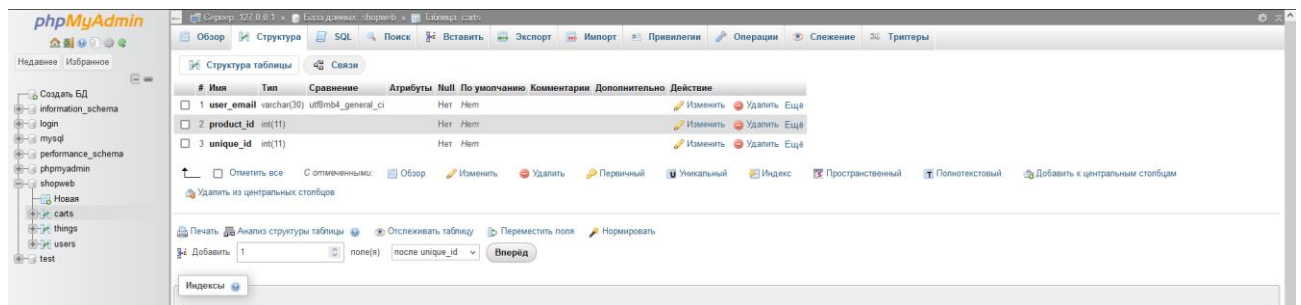


Рис.2.Как выглядит phpMyAdmin

Название базы данных – ShopWeb. В ней будет реализовано 3 таблицы:

Users – хранится user\_id (тип INT(100),дополнительно

AUTO\_INCREMENT),email(тип varchar(30)),password(тип varchar(20)).

Things- хранится id(INT(5),name(varchar(50)),price(INT(6)),img(varchar(200))

Carts-хранится user\_email(varchar(30)),product\_id(int(11)),unique\_id(int(11))

Users отвечает за данные почт и паролей аккаунтов, Things – за базу товаров, а carts – за их корзины

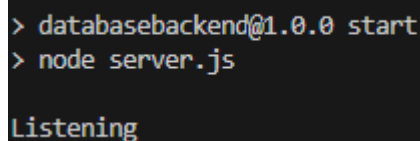
В backend-части приложения весь программный код организован в файле server.js, который выполняет функцию простого посредника между фронтендом

(frontend) и базой данных. Для эффективного и гармоничного взаимодействия компонентов решено использовать несколько популярных библиотек, таких как Express, Cors и SQL.

Библиотека Express обеспечивает удобную и гибкую обработку HTTP-запросов, позволяя легко настраивать маршруты и обрабатывать различные типы запросов от клиента. Cors (Cross-Origin Resource Sharing) используется для обработки запросов от других источников, что позволяет предотвратить проблемы с безопасностью взаимодействия между различными доменами.

SQL-библиотека используется для работы с базой данных, обеспечивая выполнение SQL-запросов к базе данных. В каждой из подстраниц приложения реализована функция для взаимодействия с базой данных, которая позволяет получать данные из базы и сохранять их, обеспечивая эффективное управление информацией и выполнение операций чтения и записи данных.

Таким образом, серверная часть приложения с серверным файлом server.js служит важным звеном между пользовательским интерфейсом и базой данных, используя для этого надежные библиотеки Express, Cors и SQL, обеспечивая стабильность и безопасность работы приложения.



```
> databasebackend@1.0.0 start
> node server.js
Listening
```

Рис.3. Запуск Backend-части приложения через командную строку.

Библиотека react-router-dom является мощным инструментом для создания многостраничных приложений на основе React. Она позволяет управлять маршрутизацией и переходами между различными страницами в приложении. Ориентированная изначально на одностраничные приложения, React благодаря react-router-dom приобретает возможность легко реализовать и масштабировать многостраничные проекты.



Библиотека Mantine, в свою очередь, предоставляет готовые базовые элементы и компоненты для веб-приложений, предварительно настроенные и стилизованные. Это упрощает процесс разработки, поскольку разработчику не приходится каждый раз создавать элементы с нуля, а можно быстро воспользоваться готовыми решениями, начиная от форм для регистрации и заканчивая выплывающими контекстными меню.

Использование библиотеки Mantine действительно позволяет значительно упростить процесс разработки веб-приложений, поскольку разработчики могут сконцентрироваться на бизнес-логике приложения, не тратя лишнее время на верстку и стилизацию базовых компонентов. Этот подход значительно увеличивает продуктивность команды, поскольку используются проверенные и готовые решения, что сокращает время разработки и повышает эффективность работы.

В случае главной страницы приложения, которая состоит из двух частей - навигационной панели и основного контента сайта, библиотека Mantine может быть очень полезна. Навигационная панель может быть легко создана и стилизована с использованием готовых компонентов Mantine, позволяя быстро и качественно реализовать функционал навигации по приложению.

Основной контент сайта также может быть оформлен с помощью компонентов из библиотеки Mantine, обеспечивая стильное и современное оформление страницы без необходимости создания стилей с нуля. Это значительно ускоряет процесс разработки и позволяет фокусироваться на функционале приложения, не отвлекаясь на мелкие детали

верстки.

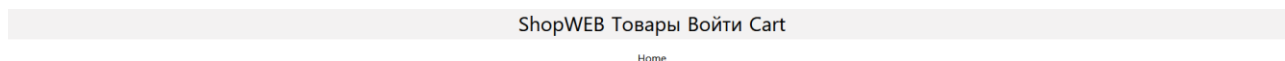


Рис.4. Прототип сайта.

Веб-приложение использует навигационную панель для обеспечения удобного перехода между различными страницами и функциональными элементами, такими как дочерние элементы корзины и вход в аккаунт. Каждый элемент панели занимается своей определенной функцией, что способствует более гибкой и простой настройке приложения. Такой подход к организации навигации значительно улучшает пользовательский опыт, обеспечивая понятность и легкость перемещения по различным разделам веб-сайта. За счет четкой структуры навигационной панели пользователи могут быстро находить необходимую информацию и использовать функционал приложения без лишних сложностей.

Веб-страница интернет-магазина является специальным интерактивным инструментом, который представляет собой удобный каталог товаров, организованный в виде сетки, способной автоматически регулироваться при любых изменениях, таких как добавление новых позиций или удаление существующих из базы данных. Каждый товар на странице оборудован кнопкой "Добавить в корзину", которая запускает соответствующий запрос на включение данного товара в корзину пользователя с назначением уникального идентификатора. Этот уникальный идентификатор необходим для корректного

определения выбранного товара и последующего отображения его в корзине, обеспечивая точное соответствие между заказами и товарами для удовлетворения потребностей пользователей.



Рис.5. Вид товара.

Для реализации корзины, которая выводит информацию из базы данных в зависимости от пользователя, можно эффективно использовать уникальные идентификаторы, такие как почта пользователя, для определения соответствующей корзины. Это позволит хранить корзины всех пользователей в одной таблице базы данных, с каждым элементом корзины, связанным с уникальным идентификатором пользователя.

Каждый элемент корзины, отображаемый на странице, должен содержать кнопку удаления, которая отправляет запрос на удаление конкретного элемента корзины по уникальному идентификатору, присвоенному данному элементу.

Это обеспечит возможность пользователям удалять конкретные товары из своей корзины с помощью соответствующей кнопки удаления.

При обработке запроса на удаление нужно удостовериться, что пользователь имеет соответствующие права доступа к данному действию, чтобы избежать несанкционированного удаления элементов корзины. Следует также обеспечить безопасность базы данных и защиту от SQL-инъекций при выполнении операций удаления.

Такой подход к реализации корзины позволит эффективно управлять содержимым корзины пользователей, обеспечивая пользовательский опыт с возможностью добавления и удаления товаров из корзины с помощью

интуитивно понятного интерфейса

Корзина

×



BOBUX SELL 24SALE 4\$

×



BOBUX SELL 24SALE 4\$

×

Рис.6. Корзина.

функция входа на сайт выполняет важную задачу - проверку наличия учетной записи в базе данных, и, при успешном входе, предоставляет доступ к различным разделам магазина и корзине, а также устанавливает куки, которые помечают пользователя как авторизованного. Это позволяет сохранить состояние сеанса даже после выхода с сайта, обеспечивая удобство клиентам при повторном посещении, так как они могут оставаться залогиненными без необходимости повторного ввода учетных

данных.

isAuth	1	localhost	/	Сессионная
--------	---	-----------	---	------------

Рис.7. куки isAuth

### Заключение:

В результате проведенного проекта мы приобрели ценный опыт работы с созданием базы данных для хранения информации о товарах для веб-приложения. Мы изучили особенности предметной области, проанализировали потребности пользователей и определили ключевые требования к структуре данных.

Проектирование и реализация физической модели данных позволили нам глубже понять взаимосвязи между сущностями товаров, их характеристиками и способами отображения на веб-приложении. Мы овладели навыками выбора и применения соответствующих инструментов разработки, а также научились эффективно структурировать данные для обеспечения оптимальной работы приложения.

Анализ результатов проекта позволил нам выявить сильные и слабые стороны разработанной базы данных и веб-приложения, а также сделать выводы о необходимости дальнейших улучшений и оптимизаций. Мы убедились в значимой роли баз данных в современных информационных технологиях, особенно в контексте решения задач бизнеса и улучшения пользовательского опыта.

Итоговый вывод подчеркивает важность грамотного проектирования и разработки баз данных для эффективного функционирования веб-приложений, а также необходимость постоянного совершенствования и адаптации к изменяющимся требованиям и потребностям пользователей.

### Памятка:

BACKEND.

Server.js:

```
const express = require('express');
const mySQL = require('mysql');
const cors = require('cors');
//database info
const db = mySQL.createConnection({
  host:"localhost",
  user:"root",
  password:"",
  database:"shopweb"
})

const app = express();
app.use(express.json());
app.use(cors({
  origin: 'http://localhost:5173', // Замените на домен вашего фронтенда
  credentials: true
}));
app.get('/things',(req,res)=>{
  const sql ="SELECT * FROM things";
  db.query(sql,function(err,data){
    if(err) throw err;
    res.send(data)
  });
});
app.post('/getCart',(req,res)=>{
  const sql ="SELECT * FROM `carts` WHERE `user_email`= ? ";
  db.query(sql,[req.body.email],function(err,data){
    if(err) throw err;
    res.send(data)});
});
app.post('/users',(req,res)=>{
  const sql = "SELECT * FROM users WHERE email = ? AND password = ?";
  db.query(sql,[req.body.email,req.body.password] ,(err,data)=>{
    if(err) return res.json("Error");
    if(data.length>0){
      return res.json("Login Successfully")
    } else {
      return res.json("No record")
    }
  })
});
app.post('/addCart',(req,res)=>{
  const sql = "INSERT INTO `carts`(`user_email`, `product_id`, `unique_id`)
VALUES (?, ?,?) ";

  db.query(sql,[req.body.email,req.body.product,parseInt(Math.random()*1000)],(err,data)=>{
    if (err){
      return res.json("Error");
    }
  })
});
```

```

        return res.json(data);
    })
})
app.post('/removeCart',(req,res)=>{

    const sql = 'DELETE FROM `cars` WHERE `unique_id` = ?';
    var id = req.body.unique_id;
    console.log(req.body.unique_id);

    db.query(sql,[id],function (error, results, fields) {
        if (error) throw error;
        console.log('deleted ' + results.affectedRows + ' rows');
        return res.json
    });
})

app.post('/reg',(req,res)=>{
    const sql = "INSERT INTO `users` (`email`,`password`) VALUES (?,?)"
    db.query(sql,[req.body.email,req.body.password],function (err,result){
        if (err) throw err;
        console.log('user added')
        return res.json
    })
})

app.listen(8081,()=>{
    console.log("Listening");
})

```

Package.json:

```

{
  "name": "databasebackend",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": "",
  "dependencies": {
    "body-parser": "^1.20.2",
    "cors": "^2.8.5",
    "express": "^4.19.2",
    "mysql": "^2.18.1"
  }
}

```

FRONTEND.

Main.jsx:

```
import React from 'react';
```



```

import ReactDOM from 'react-dom/client';
import App from './App.jsx';
import './index.css';
import '@mantine/core/styles.css';
import {MantineProvider} from '@mantine/core';
import { BrowserRouter } from 'react-router-dom';

ReactDOM.createRoot(document.getElementById('root')).render(
  <MantineProvider>
    <BrowserRouter>
      <App/>
    </BrowserRouter>
  </MantineProvider>
)

```

App.jsx:

```

import {Routes,Route} from 'react-router-dom'
import Navbar from './components/Navbar'
import Home from './pages/Home'
import './App.css'
import { Center, Container, Stack } from '@mantine/core'
import Shop from './pages/Shop';
import Register from './pages/register'
function App() {
  return (<div className="body">
    <Stack>
      <div className='nav'><Navbar/></div>
      <Routes><Route path="/" element={<Home/>}/><Route path="/shop"
element={<Shop/>}/><Route path="/register" element={<Register/>}/></Routes>

      </Stack>
    </div>)
}

export default App

```

App.css:

```

.body{
  position: fixed;
  top: 0;
  left: 0;
  bottom: 0;
  right: 0;
  background: white;
}
.nav{
  top:0;
  background: rgb(243, 242, 242);
  box-shadow: 0 -10px 0 #000;
}

```

Shop.jsx:

```

import axios from "axios";
import { useEffect, useState } from "react";
import "./Shop.css";
import { Button, Center, SimpleGrid, em } from '@mantine/core';
import Cookies from 'js-cookie';

//components
function addToCart(prod){
  axios.post("http://localhost:8081/addCart",{
    email: Cookies.get("authEmail"),
    product: Number(prod)}).then(
    (response)=>{
      console.log(response)
    });
};

function shopthing(){
  const arr = getDB()
  var renderOutput = arr.map(item =>
    <div className="MainElement" key={item.id}>
      <img src={item.img}className="shopElement" width="200" height="200"></img>
      <div className="shopElement">{item.name}</div>
      <Center>
        <div className="shopElement price">{item.price}$</div>
        <Button onClick={()=>addToCart(item.id)}>Buy</Button>
      </Center>
    </div>
  );

  return(    <SimpleGrid cols={arrLength(arr)}>    {renderOutput}
  </SimpleGrid>)
}

function arrLength(array){
  if (array.length <3){
    return array.length
  }
  else{return 3}
}

//main
function getDB(){
  const [values,setValues]=useState([]);

  axios.get('http://localhost:8081/things',{
    timeout : 10000
  })
  .then(response => {
    const data = response.data;
    const valuesArray = data.map(item=>({
      id:item.id,
      name:item.name,

```

```

        price:item.price,
        img: item.img
      )))
      setValues(valuesArray);
    })
    .catch(error=>{
      console.error(error);
    });

    return values
  }
}

function Shop(){
  if( Cookies.get("isAuth") == undefined ){
    window.location.href = '/'
    console.log("cookie does not exist")
  }
  if(Cookies.get("isAuth")==='1'){
    return(<><div>
      <Center className="pagename">Товары</Center>
      <Center> {shopthing()}</Center>
    </div>
    </>)
  }
}

export default Shop

```

Shop.css:

```

.mainGrid{
  margin: 2%;
}

.pagename{
  font-family:sans-serif;
  font-size: 25px;
}

.wrapper{
  display: grid;
  grid-template-columns: auto auto auto;
  gap: 10px;
  grid-auto-rows: minmax(100px, auto);
}

.price{
  margin-right: 20px;
}

```

Register.jsx:

```

import { TextInput,PasswordInput,Box,Button, Center } from "@mantine/core"
import { useToggle, upperFirst } from '@mantine/hooks'
import { Form, useForm } from '@mantine/form';
import axios from "axios";
function regHandler(email,password){
  axios.post("http://localhost:8081/reg",{
    email: email,

```

```

        password: password}).then(
        (response)=>{
            console.log(response)
        });
    });
}

function Register(){
    const [type, toggle] = useToggle(['login', 'register']);

    const form = useForm({
        initialValues: {
            email: '',
            name: '',
            password: '',
            terms: true,
        },

        validate: {
            email: (val) => (/^\S+@\S+$/.test(val) ? null : 'Invalid email'),
            password: (val) => (val.length <= 6 ? 'Password should include at least 6
characters' : null),
        },
    });

    return(<>
        <Box maw={340} mx="auto">
            <form onSubmit={form.onSubmit(() =>
{regHandler(form.values.email,form.values.password)}}>
                <TextInput
                    required
                    label="Email"
                    placeholder="example@mail.com"
                    value={form.values.email}
                    onChange={(event) => form.setFieldValue('email',
event.currentTarget.value)}
                    error={form.errors.email && 'Invalid email'}
                    radius="md"
                />
                <PasswordInput
                    required
                    label="Password"
                    placeholder="Your password"
                    value={form.values.password}
                    onChange={(event) => form.setFieldValue('password',
event.currentTarget.value)}
                    error={form.errors.password && 'Password should include at least 6
characters'}
                    radius="md"
                />
                <Center><Button type="submit" radius="xl">Submit
                </Button></Center>
            </form>
        </Box>
    );
}

```

```

    </>)
  }
  export default Register

```

Home.jsx:

```

import { TextInput, PasswordInput, Box, Button, Center } from "@mantine/core"
import { useToggle, upperFirst } from '@mantine/hooks'
import { Form, useForm } from '@mantine/form';
import axios from "axios";
function regHandler(email,password){
  axios.post("http://localhost:8081/reg",{
    email: email,
    password: password}).then(
    (response)=>{
      console.log(response)
    });
}

function Register(){
  const [type, toggle] = useToggle(['login', 'register']);

  const form = useForm({
    initialValues: {
      email: '',
      name: '',
      password: '',
      terms: true,
    },

    validate: {
      email: (val) => (/^\S+@\S+$/.test(val) ? null : 'Invalid email'),
      password: (val) => (val.length <= 6 ? 'Password should include at least 6
characters' : null),
    },
  });
  return(<>
    <Box maw={340} mx="auto">
      <form onSubmit={form.onSubmit(() =>
{regHandler(form.values.email,form.values.password)}}>
        <TextInput
          required
          label="Email"
          placeholder="example@mail.com"
          value={form.values.email}
          onChange={(event) => form.setFieldValue('email',
event.currentTarget.value)}
          error={form.errors.email && 'Invalid email'}
          radius="md"
        />
        <PasswordInput
          required
          label="Password"
          placeholder="Your password"

```

```

        value={form.values.password}
        onChange={(event) => form.setFieldValue('password',
event.currentTarget.value)}
        error={form.errors.password && 'Password should include at least 6
characters'}
        radius="md"
      />
      <Center><Button type="submit" radius="xl">Submit
</Button></Center>
    </form>
  </Box>
</>)
}
export default Register

```

Navbar.jsx:

```

import { Center, Flex, Grid, Popover } from '@mantine/core';
import './Navbar.css'
import {Link } from "react-router-dom";
import Login from './Login';
import Cart from './Cart';
import Cookies from 'js-cookie';
function Navbar(){
  if(Cookies.get("isAuth")==="1"){
    return (<
      <Center >
        <Flex className='nav' gap={10}>
          <Link to="/">ShopWEB</Link>
          <Link to="/shop">Товары</Link>
          <div><Login/></div>
          <div><Cart/></div>
        </Flex>
      </Center>
    </>);
  }
  else{
    return(<
      <Center >
        <Flex className='nav' gap={10}>
          <Link to="/">ShopWEB</Link>
          <div><Login/></div>
        </Flex>
      </Center>
    </>)
  }
}
export default Navbar;

```

Navbar.css:

```

.nav Link{
  user-select: none;
}

```

```

a:link {
  text-decoration: none;
  color: black;
}

a:visited {
  text-decoration: none;
  color: black;
}

a:hover {
  text-decoration: none;
  color: black;
}

a:active {
  text-decoration: none;
  color: black;
}
}

@media screen and (min-width: 1200px){
  .nav {font-size: 30px;}
}
@media screen and (max-width:1200px)
{
.nav{font-size: 2.5vw;}
}
@media screen and (max-width:758px) {
  .nav {font-size: 25px;}
}
}

```

## Login.jsx:

```

import { Popover, TextInput, Text, Button, PasswordInput } from "@mantine/core"
import './Login.css'
import { useState } from "react"
import axios from "axios";
import Cookie from "js-cookie"
import { useToggle } from "@mantine/hooks";
import React from "react";
import { Link } from "react-router-dom";
function Login(){
  const [email,setEmail] = useState('');
  const [password,setPassword] = useState('');
  const [isAuth,setIsAuth]=useState(false);
  const [drop,setDrop]=useState('');

  function HandleReload(){
    window.location.reload();
  }
  function logOut(){
    setIsAuth(false);
  }

```

```

    Cookie.remove("isAuth","0",{sameSite: 'Strict',Secure:'true'});
    Cookie.remove("authEmail");
    HandleReload()
  }
  function HandleSubmit(event){

    event.preventDefault();
    axios.post('http://localhost:8081/users',{email,password})
    .then(()=>{
      Cookie.set("isAuth","1",{sameSite: 'Strict',Secure:'true'});
      Cookie.set("authEmail",email);
      HandleReload();
    })
    .catch(err => console.log(err))
  }
  function AuthCheckDropDown(){
    if(Cookie.get("isAuth")==="1"){
      return(<>
        <div>Welcome!</div>
        <Button mt="xs"onClick={logout}>Log Out</Button>
      </>)
    }
    else{
      return(<>
        <TextInput label="Email" placeholder="example@mail.com" size="xs"
onChange={(event) => setEmail(event.currentTarget.value)}/>
        <PasswordInput label="Password" placeholder="Password" size="xs" mt="xs"
onChange={(event) => setPassword(event.currentTarget.value)}/>
        <div><div><Button mt="xs"
onClick={HandleSubmit}>Login</Button></div><div><Link to='/register'><Button
mt="xs">Register</Button></Link></div></div>
      </>)
    }
  }
  return(<>
    <Popover width={200} position="bottom" withArrow shadow="md" >
      <Popover.Target>
        <div>Войти</div>
      </Popover.Target>
      <Popover.Dropdown>
        {AuthCheckDropDown()}
      </Popover.Dropdown>
    </Popover>
  </>)
}
export default Login

```

Login.css:

```

.element{
  font-size: 50px;
}

```

Cart.jsx:

```

import { Drawer,Button, Grid, Center,Text, Flex,CloseButton } from "@mantine/core"

```



```

import { useDisclosure } from '@mantine/hooks';
import Cookies from 'js-cookie';
import { Link } from "react-router-dom";
import axios from "axios";
import { useEffect, useState } from "react";
function cartDrawer(){

    function getUserCart(){
        const [cart,setCart]=useState([])
        var email = Cookies.get('authEmail');

        axios.post('http://localhost:8081/getCart',{email:email}).then((response)=>{
            var data = response.data
            setCart(data)
        })
        return cart
    }

    function delCart(uid) {
        axios.post('http://localhost:8081/removeCart', { unique_id:uid })
        .then(() => {
            console.log(uid);
        })
        .catch(err => console.log(err));
    }

    function getDB(){
        const[values,setValues]=useState([]);

        axios.get('http://localhost:8081/things',{
            timeout : 10000
        })
        .then(response => {
            const data = response.data;
            const valuesArray = data.map(item=>({
                id:item.id,
                name:item.name,
                price:item.price,
                img: item.img
            }))
            setValues(valuesArray);
        })
        .catch(error=>{
            console.error(error);
        });
    }

```

```

        return values
    }
    var arr = getDB()
    var cart = getUserCart()
    var cart = cart.map(item=>
    {
        const product = arr[item.product_id-1];
        if(product){
            return({
                uid: item.unique_id,
                name: product.name,
                img: product.img,
                price:product.price
            })
        }
    })

    })
    var result = cart.map(item=>{return(
    <Center><Flex direction="row" gap="md" mt="sm">
        <img src={item.img} width={50} height={50}/>
        <div>{item.name}</div>
        <div>{item.price}$</div>
        <CloseButton size="x1" onClick={()=>{delCart(item.uid)}} />
    </Flex>
    </Center>))}

    )
    return(result)
}
function Cart(){
    const [opened, { open, close }] = useDisclosure(false);
    return (
        <>
            <Drawer offset={8} radius="md" opened={opened} onClose={close}
title="Корзина" position="right">
                <Flex direction="column">{cartDrawer()}</Flex>
            </Drawer>
            <Link onClick={open}>Cart</Link>
        </>);
}
export default Cart

```

### Источники:

1. Попов Д.И., Попова Е.Д. Информационные технологии. Базы данных [Текст]: учеб. пособие для студ. вузов /Д.И. Попов. – М: гос. ун-т, печати, 2009. – 117 с.
2. Карпова И.П. Базы данных [Текст]: учеб. пособие для студ. вузов /И.П. Беляев. – М: Московский государственный институт электроники и математики, 2009. – 130 с.
3. Энциклопедия [Электронный ресурс]. - <https://ru.wikipedia.org>
4. Руководство по node js [Электронный ресурс] - <https://metanit.com/web/nodejs/>
5. Руководство по React [Электронный ресурс] - <https://metanit.com/web/react/>