НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. Р. Е. Алексеева

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА к курсовой работе

Спорышев Егор Максимович

(фамилия, имя, отчество)

Институт Институт радиоэлектроники и информационных технологий

Кафедра Графические информационные системы

Группа 21-ИСТ-4-1

День защиты « 21 » декабря 2022 г.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. Р.Е. Алексеева

Кафедра Графические информационные системы

Заведующий кафедрой ГИС

_____ Филинских А.Д.
(подпись) *(фамилия и. о.)*
(дата)

Разработка сайта на Angular

«ShopAndLife»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

КОНСУЛЬТАНТЫ:

1.По

_____ _____ _____
(подпись) *(фамилия и. о.)* *(дата)*

2.По

_____ _____ _____
(подпись) *(фамилия и. о.)* *(дата)*

3.По

_____ _____ _____
(подпись) *(фамилия и. о.)* *(дата)*

РЕЦЕНЗЕНТ

_____ _____ _____
(подпись) *(фамилия и. о.)* *дата)*

РУКОВОДИТЕЛЬ:

_____ Юматов М.А.
(подпись) *(фамилия и. о.)*
(дата)

СТУДЕНТ

Спорышев Е.М.
(подпись) *(фамилия и. о.)*
21-ИСТ-4-1
(дата) *(группа или шифр)*

Проект защищен _____

Протокол № _____

С оценкой _____

					КР – НГТУ – 090302 – 21-ИСТ-4-1 – 16 – ПЗ		
Изм.	Лист	№ докум.	Подпись	Дата			
Провер.		Юматов М.А.			ПОЯСНИТЕЛЬНАЯ ЗАПИСКА	Лит.	Лист 2 / Листов 46
Разраб.		Спорышев Е.М.					Кафедра ГИС гр. 21-ИСТ-4-1

Нижегородский государственный технический университет им. Р.Е. Алексеева

Кафедра: Графические информационные системы

УТВЕРЖДАЮ
Зав. кафедрой **Филинских А.Д.**

З А Д А Н И Е
на курсовое проектирование

Студент: Спорышев Егор Максимович, группа 21-ИСТ-4-1

Тема курсового проекта: Разработка Web-приложения «ShopAndLife»

Исходные данные к проекту: основные знания и сведения о Typescript как основном языке написания работы, сведения об Angular, Node.js, Webpack и json-server

Содержание графического материала:

чертежи:
1. Иллюстрации с примером оформления страниц сайта
2. Картинки из интернета

Содержание пояснительной записки:

Перечень вопросов, подлежащих разработке

Анализ исходных данных, и разработка ТЗ

Руководство пользователя

Руководство программиста

Листинг

Основная рекомендуемая литература:				
[1] Руководство по Angular 14. Режим доступа https://metanit.com/web/angular2/				
[2] Angular. Режим доступа https://angular.io/docs				
			Руководитель	Юматов М.А.

«21» декабря 2022 г.

Студент ФИО Спорышев Егор Максимович

Дата 21.12.2022

Содержание

Введение

В современном мире огромное значение приобретают навыки пользования web-приложениями, с помощью них люди могут хранить данные о прочитанных книгах, просмотренных фильмах, о событиях, которые они хотят посетить и т.д., получать актуальную информацию об интересующих их событиях, просматривать различные статьи и оставлять комментарии на волнующие их темы. Т.е. web-приложения должны максимально удовлетворять информационным потребностям людей.

В последнее время наибольшую популярность набирают маркетплейсы и это неудивительно, ведь возможность просмотреть все товары сразу и осуществить покупку необходимого экономит огромное количество времени покупателей. Более того, продавцам также приходится прилагать гораздо меньше усилий для продажи собственных товаров. Именно поэтому и была выбрана тема интернет-магазина.

Сегодня самым распространенным типом сайтов являются, конечно же, динамические сайты, в которых генерация страниц осуществляется на стороне сервера. При всех очевидных удобствах для разработчиков (каждый запрос создаёт страницу с чистого листа), такие сайты порой представляют собой настоящую головную боль для клиентов. Подход SPA во многих случаях оказывается гораздо более эффективным.

С точки зрения этого подхода сайт понимается не как набор страниц, а как набор состояний одной и той же HTML-страницы. При смене состояния происходит асинхронная подгрузка нового контента без перезагрузки самой страницы.

SPA представляет собой не сайт в классическом понимании, а приложение, которое исполняется на стороне клиента, в браузере. Поэтому с точки зрения пользователя проблем со скоростью работы почти не бывает даже при медленном или нестабильном соединении с Интернетом (например, при просмотре сайта с мобильного устройства). Высокая скорость работы обеспечивается ещё и благодаря тому, что с сервера на клиента приходит теперь не разметка, а данные (в основном в формате JSON), и размер их невелик.

1. Анализ исходных данных, и разработка ТЗ

1.1. Назначение разработки

Разработано приложение – каталог товаров “ShopAndLife”, которое предназначено для просмотра существующих товаров и добавления собственных, а также их редактирования.

1.2. Минимальные требования к составу и параметрам технических средств: ЭВМ, внешние устройства

Функционирование приложения обеспечивается следующими параметрами технических средств:

- операционная система Windows (начиная с 7 версии);
- 2 Гб RAM и более;
- CPU с тактовой частотой не менее 2.3ГГц;
- монитор;
- устройство для ввода информации;
- доступ в сеть Интернет;
- Любой современный браузер с поддержкой TypeScript, HTML и CSS.

1.3. Требования к информационной и программной совместимости

Для успешного дальнейшего выполнения курсовой работы требовалось установить необходимое ПО:

- Node.js — это кроссплатформенная среда с открытым исходным кодом для разработки серверных и сетевых приложений;
- Webpack — это статический модульный сборщик для приложений на JavaScript;
- Angular и Angular CLI — это фреймворк от компании Google для создания клиентских приложений, который нацелен на разработку одностраничных приложений (1);
- TypeScript — это расширенная версия языка JavaScript, главной целью которого является упрощение разработки крупных JS-приложений;
- WebStorm— мощная IDE для веб-разработки;
- json сервер.

1.4. Требования к функциональным характеристикам

В системе «ShopAndLife» будут реализованы следующие возможности:

- Регистрация и авторизация в системе;
- Просмотр существующих товаров;
- Добавление собственных товаров;
- Добавление товаров в корзину, а также удаление их из нее;
- Оформление заказа;
- Просмотр данных своего аккаунта;
- Поиск товаров;
- Возможность перехода в соц. Сети и мессенджеры;
- Сохранение данных о пользователе и продуктов.

1.5. Выбор и обоснование языков программирования и используемых инструментальных средств

Для реализации данного приложения мною был задействован язык программирования TypeScript, который имеет ряд преимуществ над JavaScript – например, поддержка статической типизации, лучшая поддержка со стороны IDE. Доступ к новым возможностям ECMAScript. В TypeScript есть поддержка новых возможностей ECMAScript, поэтому можно разрабатывать приложения с помощью новейших инструментов, не волнуясь о поддержке их браузером. В качестве среды разработки – WebStorm. WebStorm позволяет автоматизировать рутинную работу и легко справляться со сложными задачами, делая разработку более увлекательной.
Для упрощения работы был подключен CCS - фреймворк – Tailwind.

2. Внешняя спецификация

2.1. Разработка сематического ядра

Семантическое ядро – ключевые слова, по которым пользователь может найти данное приложение.

Products (551 тыс. показов в месяц), shopping (1 млн. показов в месяц), buy (582 тыс. показов в месяц), sell (187 тыс. показов в месяц).

2.2. Разработка структуры приложения (карта)

Создаваемое реактивное приложение будет состоять из разделов:

1. Enter (вход)
2. Registration (регистрация)
3. Products (продукты)
4. Basket (корзина)
5. About (о нас)
6. Contacts (контакты)
7. Account (личный кабинет)

С главной страницы (Products) можно перейти в раздел About и Contacts, а также войти через свой аккаунт или зарегистрироваться (Enter). Внутри системы можно добавить собственные товары. Существующие товары можно добавлять в корзину и оформлять заказ. Также можно перейти в раздел Account, чтобы посмотреть данные собственного аккаунта. Внутри основной системы присутствуют разделы About и Contacts. Из аккаунта можно выйти, нажав на поле Exit.

.

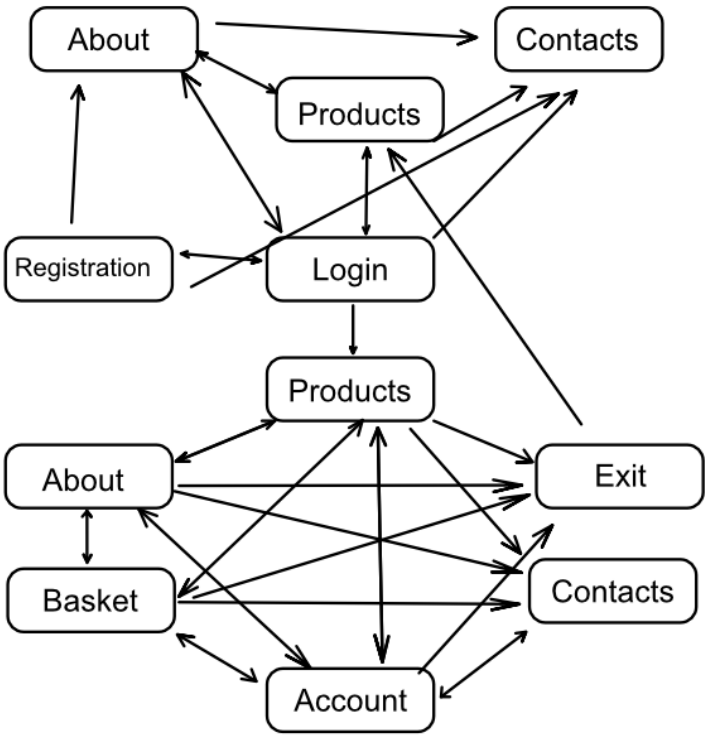

Рисунок 1 – Карта сайта

3. Руководство пользователя

3.1. Назначение программы

Разработанное приложение предназначено для просмотра существующих товаров и их покупки. А также для добавления собственных.

3.2. Описание интерфейса

Требования к интерфейсу:

- интерфейс пользователя должен быть удобным, понятным и простым в использовании;
- Работу пользователя должен обеспечивать с высокой скоростью;
- обеспечение защиты от человеческих ошибок;

После загрузки приложения открывается главная страница со всеми товарами(рисунок 2), а также меню навигации, с помощью которого можно перейти на страницу *About*, чтобы узнать о компании. При наведении курсора на

поле *Contacts*, пользователь увидит выпадающий список с разными соц. сетями компании, на которые можно перейти. О каждом о товаре можно узнать доп. информацию, нажав на кнопку Show Details. Также можно осуществлять поиск необходимых товаров с помощью поля ввода данных Filter Product.

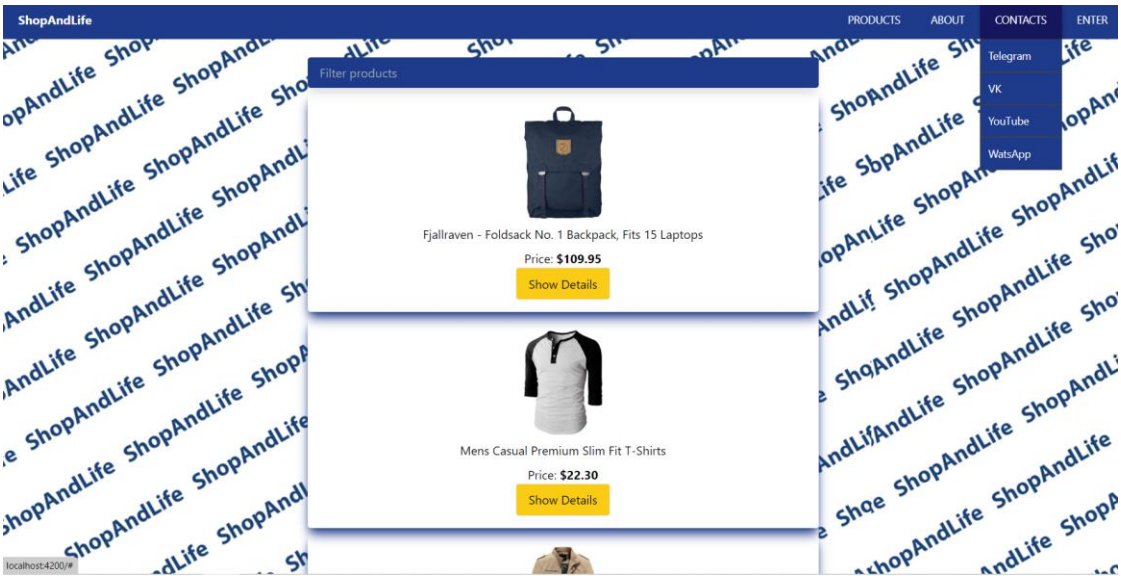

Рисунок 2 – Products

С помощью поля Enter можно войти в систему. Если аккаунта еще нет, то нажав на надпись Sign up, можно зарегистрироваться (рисунок 3, рисунок 4).

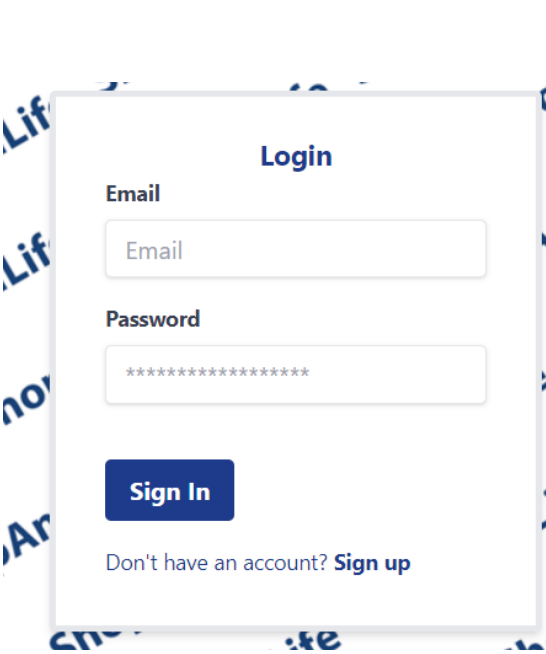

Рисунок 3 - Вход

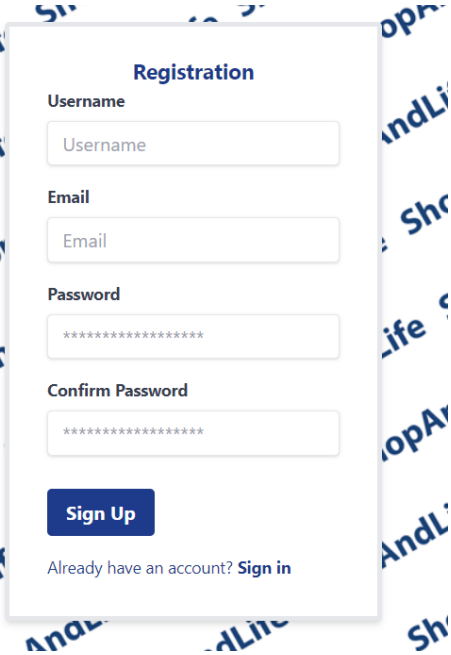

Рисунок 4 - Регистрация

После входа в систему можно добавлять собственные товары. Для этого следует нажать на круглую кнопку в правом нижнем углу приложения и заполнить необходимые поля (рисунок 5, рисунок 6).

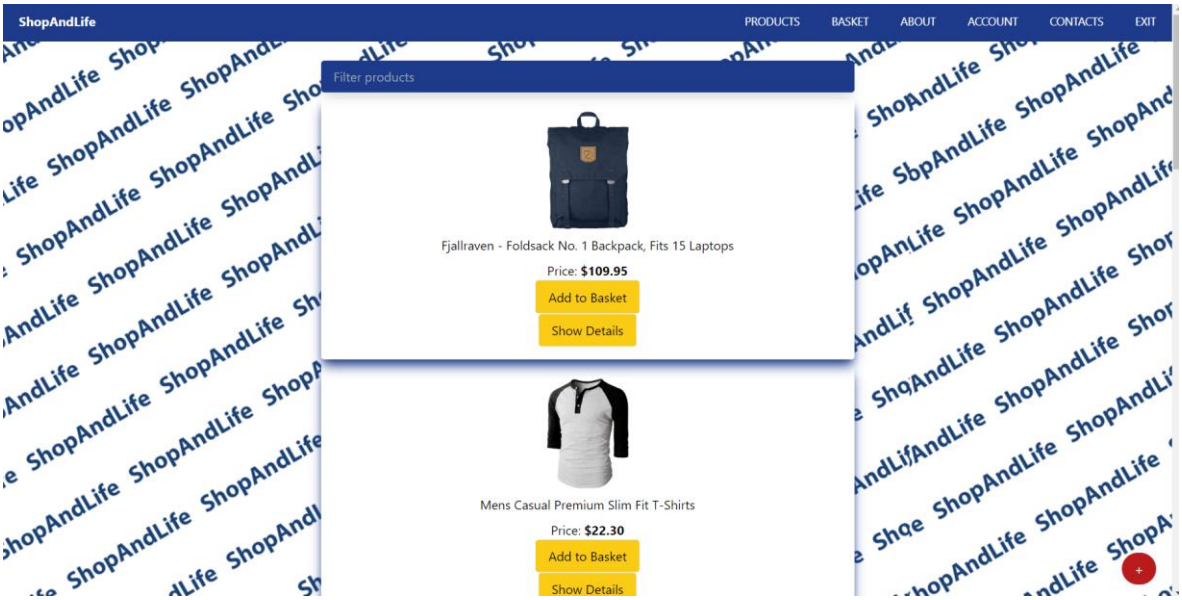

Рисунок 5 – Products (после входа в систему)

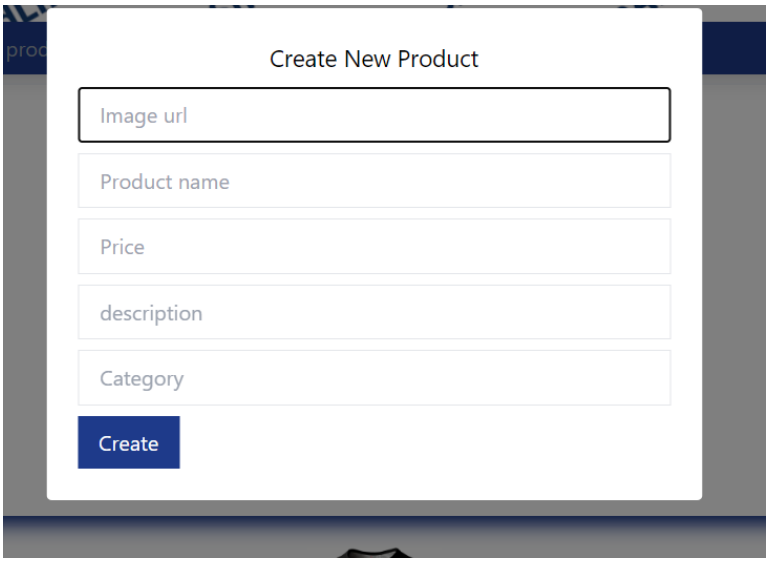

Рисунок 6 – Создание продукта

Товары можно добавить в корзину, для этого нужно использовать одноименную кнопку. Перейти к корзине можно через меню навигации (рисунок 7).

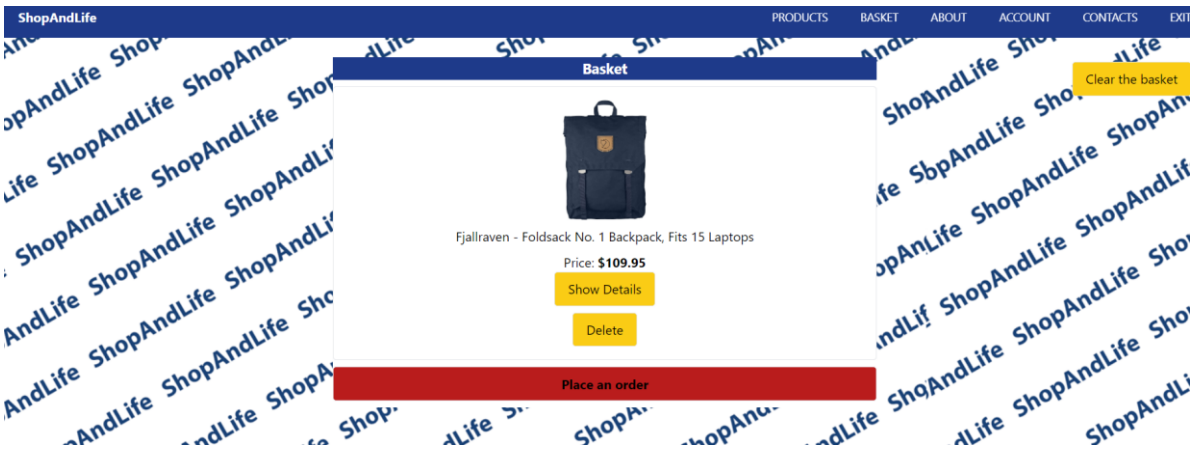

Рисунок 7 - Корзина

В поле Account можно увидеть данные, которые заполнял пользователь при регистрации (рисунок 8)

Рисунок 8 - Аккаунт

Выйти из аккаунта можно, нажав на кнопку Enter в правом верхнем углу.

3.3 Требования к входным данным

При авторизации поле Email не должно быть пустым и должно содержать корректный формат ввода почты. Пароль не должен быть пустым.

При регистрации поле Email не должно быть пустым, должно содержать корректный формат ввода почты и не должно содержать ранее введенный дру-

гим пользователем email. Длина пароля должна быть больше 8, подтверждающий пароль должен совпадать с изначальным паролем. Минимальная длина Username – 3. Все поля должны быть заполненными (рисунок 9, рисунок 10).

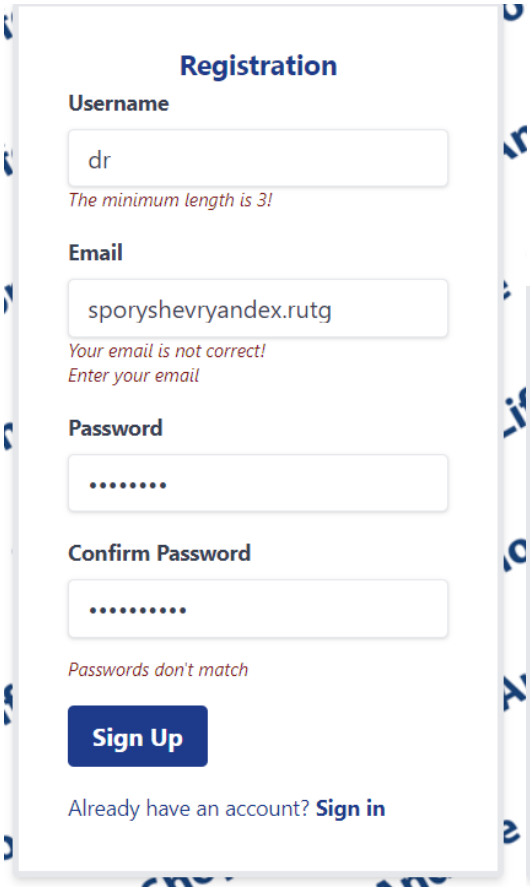

Рисунок 9 – регистрация

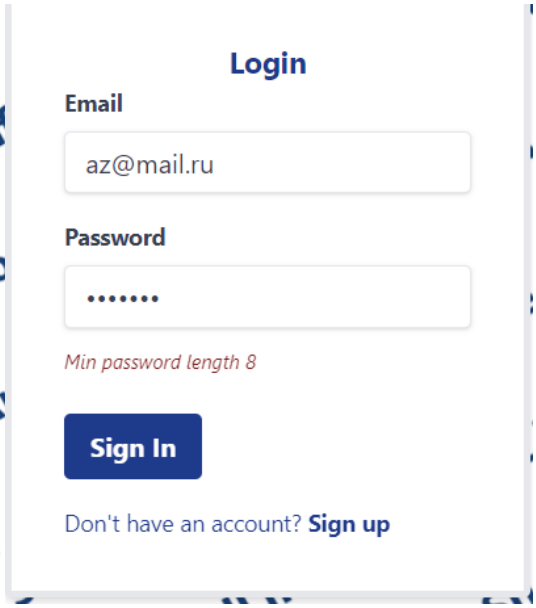

Рисунок 10 - вход

4. Руководство программиста

4.1. Организация ввода данных в программу и вывода

Основными элементами на страницах нашей системы являются формы. Реактивные формы (Angular reactive forms) построены на основе механизма, использующего реактивный подход к программированию Создание и валидация форм осуществляется прямо в компоненте, что делает работу с данными более гибкой .

Информация, используемая в системе, хранится в db.json – файле БД, в котором хранится массив пользователей, продуктов и заказов.

Вывод данных осуществляется с помощью, директивы ngFor и get запроса.

Login

```
<div id="login-container" class="flex container mt-[50px] mx-auto items-cen-
ter justify-center">
  <div class="w-full max-w-xs">
    <form class="border-4 bg-white shadow-md rounded px-8 pt-6 pb-8 mb-4"
[formGroup]="form" (ngSubmit)="onSubmit()">
      <h1 class ="font-bold mb 2 text-center text-lg text-blue-
900">Login</h1>
      <div class="mb-4">
        <label class="block text-gray-700 text-sm font-bold mb-2" >
          Email
        </label>
        <input [ngClass]="email" type="email" formControlName="email"
id="email" class="shadow appearance-none border rounded w-full py-2 px-3
text-gray-700 leading-tight focus:outline-none focus:shadow-outline" place-
holder="Email" >
        <p *ngIf="email.errors?.email && email.touched" class="text-red-900
text-xs italic">Your email is not correct!</p>
        <p *ngIf="email.errors?.required && email.touched" class="text-red-
900 text-xs italic">Enter email</p>
      </div>
      <div class="mb-6">
        <label class="block text-gray-700 text-sm font-bold mb-2" >
          Password
        </label>
        <input [ngClass]="password" type="password" formControlName="pass-
word" class="shadow appearance-none border rounded w-full py-2 px-3 text-
gray-700 mb-3 leading-tight focus:outline-none focus:shadow-outline"
type="password" placeholder="******************" >
        <p *ngIf="password.errors?.minlength && password.touched"
class="text-red-900 text-xs italic">Min password length 8</p>
        <p *ngIf="password.errors?.required && password.touched" class="text-
red-900 text-xs italic" >Enter password</p>
        <p *ngIf="!check" class="text-red-900 text-xs italic" >Wrong email or
password</p>
      </div>
      <div class="flex items-center justify-between">
        <button class="bg-blue-900 hover:bg-blue-700 text-white font-bold py-
```

```
2 px-4 rounded focus:outline-none focus:shadow-outline" type="submit"
(click)="onSubmit()">
          Sign In
        </button>
      </div>
      <div class="flex items-center justify-between mt-4">
        <p class="inline-block align-baseline text-sm text-blue-900">Don't
have an account?
          <a class="inline-block align-baseline font-bold text-sm text-blue-
900 hover:text-blue-800" routerLink="/register">
            Sign up
          </a>
        </p>
      </div>
    </form>

  </div>

</div>
```

Registration

```
<div id="register-container" class="flex container mt-[50px] mx-auto items-
center justify-center">
  <div class="w-full max-w-xs">
    <form class="border-4  bg-white shadow-md rounded px-8 pt-6 pb-8 mb-4"
[formGroup]="form" (ngSubmit)="onSubmit()">
      <h1 class ="font-bold mb 2 text-center text-lg text-blue-900">Registra-
tion</h1>
      <div class="mb-4">
        <label class="block text-gray-700 text-sm font-bold mb-2" >
          Username
        </label>
        <input [ngClass]="login" type="text" formControlName="login"
id="login" class="shadow appearance-none border rounded w-full py-2 px-3
text-gray-700 leading-tight focus:outline-none focus:shadow-outline" place-
holder="Username">
        <p *ngIf="login.errors?.minlength && login.touched" class="text-red-
900 text-xs italic"> The minimum length is 3!</p>
        <p *ngIf="login.errors?.required && login.touched" class="text-red-
900 text-xs italic">Enter username</p>
      </div>
      <div class="mb-4">
        <label class="block text-gray-700 text-sm font-bold mb-2"
for="email">
          Email
        </label>
        <input [ngClass]="email" type="email" formControlName="email"
id="email" class="shadow appearance-none border rounded w-full py-2 px-3
text-gray-700 leading-tight focus:outline-none focus:shadow-outline" place-
holder="Email" >
        <p *ngIf="email.errors?.email && email.touched" class="text-red-900
text-xs italic">Your email is not correct!</p>
        <p *ngIf="email.errors?.email && email.touched"class="text-red-900
text-xs italic">Enter your email</p>
      </div>
      <div class="mb-4">
        <label class="block text-gray-700 text-sm font-bold mb-2" >
          Password
        </label>
        <input [ngClass]="password" type="password" formControlName="pass-
word"class="shadow appearance-none border rounded w-full py-2 px-3 text-gray-
```

```
700 leading-tight focus:outline-none focus:shadow-outline" place-
holder="******************">
        <p *ngIf="password.errors?.minlength && password.touched"
class="text-red-900 text-xs italic">The minimum length is 8!</p>
        <p *ngIf="password.errors?.required && password.touched"class="text-
red-900 text-xs italic">Enter your password</p>
      </div>
      <div class="mb-4">
        <label class="block text-gray-700 text-sm font-bold mb-2" for="con-
firm-password">
          Confirm Password
        </label>
        <input [ngClass]="password" type="password" formCon-
trolName="confPass" class="shadow appearance-none border rounded w-full py-2
px-3 text-gray-700 mb-3 leading-tight focus:outline-none focus:shadow-out-
line" id="confirm-password" type="password" place-
holder="******************" >
        <p *ngIf="!check" class="text-red-900 text-xs italic">Passwords don't
match</p>
      </div>
      <div class="flex items-center justify-between">
        <button class="bg-blue-900 hover:bg-blue-700 text-white font-bold py-
2 px-4 rounded focus:outline-none focus:shadow-outline" type="submit"
(click)="onSubmit()">
          Sign Up
        </button>
        <p *ngIf="emailCheck" class="text-red-900 text-xs italic text-cen-
ter">A user with the same email already exist</p>
        <p *ngIf="loginCheck" class="text-red-900 text-xs italic text-cen-
ter">A user with the same name already exist</p>
      </div>
      <div class="flex items-center justify-between mt-4">
        <p class="inline-block align-baseline text-sm text-blue-900">Already
have an account?
          <a class="inline-block align-baseline font-bold text-sm text-blue-
900 hover:text-blue-800" routerLink="/login">
            Sign in
          </a>
        </p>
      </div>
    </form>


  </div>
</div>
```

Вывод данных

```
<div *ngIf="productsService.products">
  <app-product
    *ngFor="let product of productsService.products| filterProducts: term;
let i=index"
    [product]="product"
  ></app-product>
</div>
```

4.2. Описание шаблона web-приложения (HTML5/ SCSS)

В данной работе используется несколько шаблонов:

app.component.html

В самом главном шаблоне приложения app.component.html прописываем <router-outlet></router-outlet> А также сервисы которые должны работать во всех разделах приложения.

login.component.html и registration.comonent.html

В шаблонах авторизации и регистрации прописаны формы для авторизации и регистрации, связанные их с соответствующими компонентами, так же делаем валидаторы, для корректного ввода имени, email и пароля, и блокировки формы, при некорректном вводе.

account.component.html

Этот шаблон отвечает за отображение данных аккаунта пользователя

```
<div class="space">
  <div class="tbl">
    <div class="profile">Profile</div>
    <table>
      <tr>
        <td>Login</td><td><div class="bor">{{userLogin}}</div></td>
      </tr>
      <tr>
        <td>Email</td><td><div class="bor">{{userEmail}}</div></td>
      </tr>
      <tr>
        <td>Password</td><td><div class="bor">{{userPassword}}</div></td>
      </tr>
    </table>
  </div>
</div>
```

order.component.html и order-page.component.html

Эти шаблоны отвечают за отображение товаров в корзине

```
<div [ngClass]="{
'fixed':ording
}">
  <h1 class="font-bold mb-2 text-center bg-blue-900 text-white text-
lg">{{ title | titlecase}} {{empty?'is emty':''}}</h1>
</div>
<div *ngIf="productsService.orders">
  <app-order *ngFor="let order of productsService.orders" [order]="or-
der"></app-order>
</div>
<button class="border py-2 px-4 w-full text-center rounded bg-red-700 mx-auto
font-bold mb-2 rounded" *ngIf="!empty" (click)="ording=true">Place an or-
der</button>
```

```
<button
  *ngIf="!empty"
  class="border py-2 px-4 bg-yellow-400 rounded fixed right-5 top-20 py-2 px-
4"
  (click)="removeFullOrder()"
>Clear the basket</button>
<div  class="bg-black/50 fixed top-0 right-0 bottom-0 left-0" *ngIf="ording
|| ordered" (click)="ording=false; ordered=false"></div>
<div *ngIf="ording" class="items-center">
  <form class="w-[360px] absolute rounded top-40 p-6 bg-white left-1/2 -
translate-x-1/2 fixed items-center ">
    <h1 class="text-center font-bold">Are you sure you want to place an or-
der?</h1>
    <button (click)="ording = false; doOrder(); ordered = true" class="mt-6
border py-2 px-4 rounded bg-yellow-400 flex mx-auto">Confirm</button>
  </form>
</div>
```

```
<div class="py-4 px-6 border rounded flex items-center flex-col mb-2">
  <img [src]="order.image" [alt]="order.title" class="w-1/6 mb-2">
  <h2 class="mb-2 text-md text-center">{{ order.title }}</h2>
  <p>Price: <span class="font-bold">{{order.price | currency }}</span></p>


  <button
    class="border py-2 px-4 rounded"
    [ngClass]="{
      'bg-yellow-400': !details,
      'bg-red-700': details
    }"
    (click)="details = !details"
  >
    {{ details ? 'Hide' : 'Show' }} Details
  </button>



  <div *ngIf="details">
    <p>{{ order.description }}</p>
  </div>
  <button class="border py-2 px-4 rounded bg-yellow-400 mt-2" (click)="re-
moveFromOrder()">Delete</button>
</div>
```

navigation.component.html

Шаблон, отвечающий за меню навигации

```
<nav class=" top-0 right-0 left-0 h-[50px] flex justify-between items-center
text-white bg-blue-900 fixed">
  <h1 class="font-bold px-5" >ShopAndLife</h1>

  <span class="menu"  *ngIf="!authorized">
    <ul>
      <li><a routerLink="/" class="rounded p-6">Products</a></li>
      <li><a routerLink="/about" class="rounded p-6">About</a></li>
      <li><a href="#" class="rounded p-6"><i></i>Contacts</a>
        <ul>
          <li class="bg-blue-900 "><a href="https://web.telegram.org/">Tele-
gram</a></li>
          <li class="bg-blue-900 "><a
href="https://vk.com/?ysclid=lbapku87wo227446716">VK</a></li>
```

```
          <li class="bg-blue-900 "><a
href="https://ru.youtube.com/">YouTube</a></li>
          <li class="bg-blue-900 "><a
href="https://www.whatsapp.com/">WatsApp</a></li>
        </ul>
      </li>
      <li><a routerLink="/login" class="rounded p-6">Enter</a></li>
    </ul>

  </span>
  <span class="menu"  *ngIf="authorized">
    <ul>
      <li><a routerLink="/" class="rounded p-6">Products</a></li>
      <li><a routerLink="/orders" class="rounded p-6">Basket</a></li>
      <li><a routerLink="/about" class="rounded p-6">About</a></li>
      <li><a routerLink="/account" class="rounded p-6">Account</a></li>
      <li><a href="#" class="rounded p-6"><i></i>Contacts</a>
        <ul>
          <li class="bg-blue-900 "><a href="https://web.telegram.org/">Tele-
gram</a></li>
          <li class="bg-blue-900 "><a
href="https://vk.com/?ysclid=lbapku87wo227446716">VK</a></li>
          <li class="bg-blue-900 "><a
href="https://ru.youtube.com/">YouTube</a></li>
          <li class="bg-blue-900 "><a
href="https://www.whatsapp.com/">WatsApp</a></li>
        </ul>
      </li>
      <li><a routerLink="/" class="rounded p-6" (click)="log-
out()">Exit</a></li>

    </ul>

  </span>
</nav>-->
```

product.component.html и product-page.component.html

Шаблоны главной страницы, отвечающие за отображение товаров

```
<div class="py-4 px-6 shadow-lg shadow-blue-900 rounded flex items-center
flex-col mb-2">
  <img [src]="product.image" [alt]="product.title" class="w-1/6 mb-2">
  <h2 class="mb-2 text-md text-center">{{ product.title }}</h2>
  <p>Price: <span class="font-bold">{{ product.price | currency }}</span></p>


    <button
      *ngIf="authorized"
      class="border py-2 px-4 rounded bg-yellow-400"
      (click)="ord=true; onSubmit()"

    >
      Add to Basket
    </button>


  <button
    class="border py-2 px-4 rounded"
    [ngClass]="{
      'bg-yellow-400': !details,
      'bg-red-700': details
```

```
    }"
    (click)="details = !details"
  >
    {{ details ? 'Hide' : 'Show' }} Details
  </button>



  <div *ngIf="details">
    <p>{{ product.description }}</p>
  </div>


</div>
```

```
<input
  type="text"
  class="bg-blue-900 text-white mb-2 shadow-lg rounded  py-2 px-4 w-full"
  placeholder="Filter products"
  [(ngModel)]="term"
>



<p *ngIf="loading" class="text-center text-lg">Loading...</p>



<div *ngIf="productsService.products">
  <app-product
    *ngFor="let product of productsService.products| filterProducts: term;
let i=index"
    [product]="product"
  ></app-product>
</div>

<app-modal *ngIf="modalService.isVisible$ | async" title="Create new prod-
uct">
  <app-create-product></app-create-product>
</app-modal>
```

4.3. Структура программы.

Проект состоит из главной директории app.
Директива components состоит из директив:

- create-products;
- global-error;
- modal;
- navigation;
- order;
- product;

Директива ditrectives содержит
focus.directive.ts
Директива models содержит

- order.ts;
- product.ts;
- user.ts;

Директива components состоит из директив:

- about-page;
- account-page;
- login-page;
- order-page;
- register-page;
- product-page;

Директива pipes содержит
filter-product.pipe.ts

Директива services состоит из директив:

- auth.service.ts;
- error.service.ts;
- modal.service.ts;
- product.service.ts;
- user.service.ts;

5. Тестовый пример

Протестируем страницы Логина и регистрации. При входе с незарегистрированного аккаунта вход невозможен (рисунок 11), а после регистрации удается войти (рисунок 12, рисунок 13, рисунок 14).

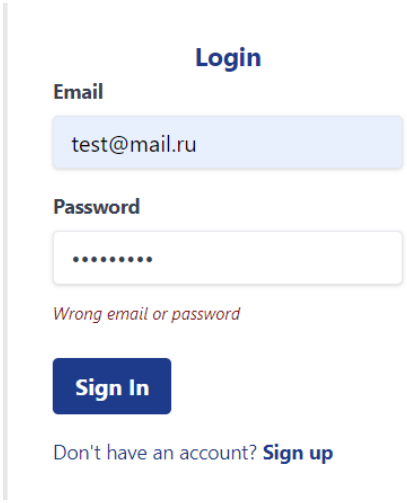

Рисунок 11 – неуспешный вход

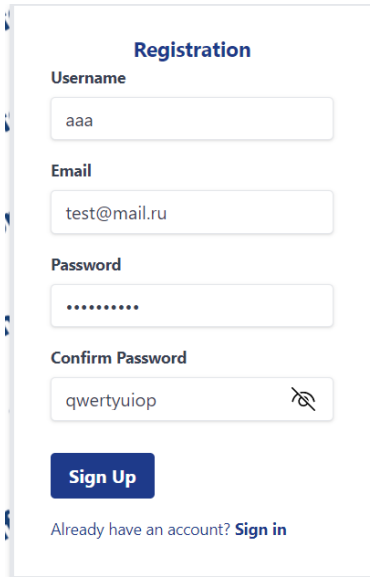

Рисунок 12 - успешная регистрация

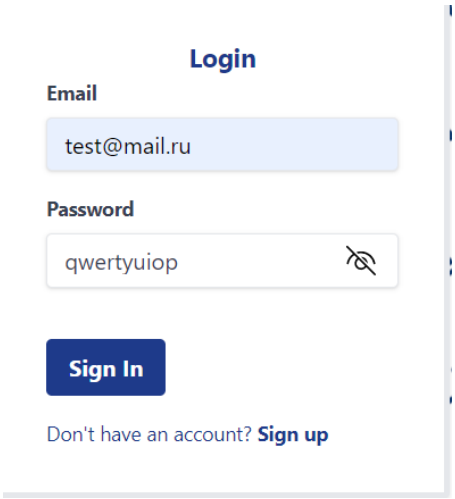

Рисунок 13 – успешный вход

Рисунок 14 – сохранение в бд.

Также протестируем создание товара (рисунок 15, рисунок 16, рисунок 17)

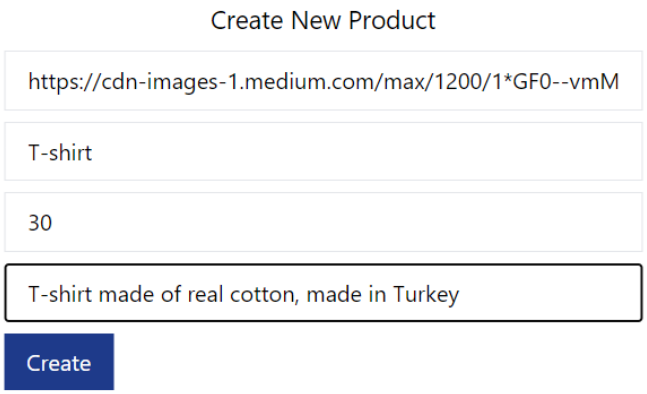

Рисунок 15 – создание товара

Рисунок 16 – созданный товар

Рисунок 17 – сохранение в бд

СПИСОК ЛИТЕРАТУРЫ

1) *Руководство по Angular 14*. Получено 28.11.2022 г., из https://metanit.com/web/angular2/
2) *json-server*. (б.д.). Получено 24.11.2022 г., из GitHub json-server: https://github.com/typicode/json-server
3) (на языке оригинала), *Angular*. Получено 24.11.2022 г., из https://angular.io/
4) *Учебно-методическое пособие к выполнению лабораторных работ для студентов направления 09.03.02 - Информационные системы и технологии.* (2022). Нижний Новгород: Нижегородский государственный технический университет им. Р.Е. Алексеева (кафедра ГИС).

ПРИЛОЖЕНИЕ

create-product.component.ts

```
import { Component, OnInit } from '@angular/core';
import {FormControl, FormGroup, Validator, Validators} from "@angular/forms";
import {ProductsService} from "../../services/products.service";
import { ModalService } from "../../services/modal.service";
import { v4 as uuidv4 } from 'uuid';

@Component({
  selector: 'app-create-product',
  templateUrl: './create-product.component.html',
  styleUrls: ['./create-product.component.scss']
})
export class CreateProductComponent implements OnInit {

  constructor(public productService: ProductsService, private modalService:
ModalService) {}
  form = new FormGroup({
    url: new FormControl<string>('', [
      Validators.required
    ]),
    title: new FormControl<string>('', [
      Validators.required,
      Validators.minLength(3)
    ]),
    price: new FormControl<number>(0, [
      Validators.required
    ]),
    description: new FormControl<string>('', [
      Validators.required
    ]),
    category: new FormControl<string>('', [
      Validators.required
    ]),
  })

  get url() {
    return this.form.controls.url as FormControl
  }

  get title() {
    return this.form.controls.title as FormControl
  }

  get price() {
    return this.form.controls.price as FormControl
  }

  get description() {
    return this.form.controls.description as FormControl
  }

  get category() {
    return this.form.controls.category as FormControl
  }

  submit() {
```

```
    this.productService.create({
      title: this.form.value.title as string,
      price: this.form.value.price as number,
      description: this.form.value.description as string,
      image: this.form.value.url as string,
      category: this.form.value.category as string,
      id: uuidv4()
    }).subscribe(() => {//добавляем слушателей
      this.modalService.close()
    })
  }

  perm = true;
  userLogin: string;
  ngOnInit(): void {
    // @ts-ignore
    const userId =  JSON.parse(localStorage.getItem('user'),
      (key, value) => (typeof value === 'object') && key !== '' ? new
userId(value.login) : value);
    this.userLogin = userId.login;
  }
}
```

global-error-component.ts

```
import { Component, OnInit } from '@angular/core';
import {ErrorService} from "../../services/error.service";

@Component({
  selector: 'app-global-error',
  templateUrl: './global-error.component.html',
  styleUrls: ['./global-error.component.scss']
})
export class GlobalErrorComponent implements OnInit {

  constructor(public errorService: ErrorService) { }

  ngOnInit(): void {
  }

}
```

modal.ts

```
import {Component, Input, OnInit} from '@angular/core';
import {ModalService} from "../../services/modal.service";

@Component({
  selector: 'app-modal',
  templateUrl: './modal.component.html',
  styleUrls: ['./modal.component.scss']
})
export class ModalComponent implements OnInit {

  @Input() title:string

  constructor(public modalService: ModalService) { }

  ngOnInit(): void {
  }}
```

navigation.component.ts

```
import { Component, OnInit } from '@angular/core';
import {AuthService} from "../../services/auth.service";
import {Router} from "@angular/router";
import {delay} from "rxjs";

@Component({
  selector: 'app-navigation',
  templateUrl: './navigation.component.html',
  styleUrls: ['./navigation.component.scss']
})
export class NavigationComponent implements OnInit {
  userLogin: string;
  userEmail: string;
  authorized = false;
  constructor(private authService:AuthService, private router : Router) { }

  ngOnInit(): void {
    try {
      // @ts-ignore
      const userId = JSON.parse(localStorage.getItem('user'),
        (key, value) => (typeof value === 'object') && key !== '' ? new
userId(value.login, value.email) : value);
      this.userLogin = userId.login;
      this.userEmail = userId.email;
      if (this.userLogin.length > 0) this.authorized = true;
    }
    catch (IOEException) {}
  }
  logout(): void {
    this.authService.ilogout();
    this.authorized=false
    this.ngOnInit()
    location.replace("/");
  }
}
```

order.component.ts

```
import {Component, Input, OnInit, Output} from "@angular/core";
import {IProduct} from "../../models/product";
import {ProductsService} from "../../services/products.service";
import {FormControl, FormGroup, Validators} from "@angular/forms";
import {ModalService} from "../../services/modal.service";
import {IOrder} from "../../models/order";

@Component({
  selector: 'app-order',
  templateUrl: './order.component.html'
})

export class OrderComponent implements OnInit{
  @Input() order:IOrder
    details = false
  userLogin: string;
  userEmail: string;
  id:number | undefined
  constructor(private productService:ProductsService, public modalService:
ModalService) {
```

```
  }
  ngOnInit(): void {
    // @ts-ignore
    const userId =  JSON.parse(localStorage.getItem('user'),
      (key, value) => (typeof value === 'object') && key !== '' ? new
userId(value.login, value.email) : value);
    this.userLogin = userId.login;
    this.userEmail = userId.email;

  }
  removeFromOrder():void{
    this.id = this.order.id
    console.log(this.order.id)
    // @ts-ignore
    this.productService.removeOrder(this.id).subscribe(location.reload())
  }
}
```

product.component.ts

```
import {Component, Input, OnInit, Output} from "@angular/core";
import {IProduct} from "../../models/product";
import {ProductsService} from "../../services/products.service";
import {FormControl, FormGroup, Validators} from "@angular/forms";
import {ModalService} from "../../services/modal.service";

@Component({
  selector: 'app-product',
  templateUrl: './product.component.html'
})

export class ProductComponent implements OnInit{
  @Input() product: IProduct
  details = false
  userLogin: string;
  userEmail: string;
  authorized = false
  id: number;
  ord = false
  ind: number = 0;
  count: string;
  form: FormGroup
  constructor(private productService:ProductsService, public modalService:
ModalService) {
  }
  ngOnInit(): void {
    // @ts-ignore
    const userId =  JSON.parse(localStorage.getItem('user'),
      (key, value) => (typeof value === 'object') && key !== '' ? new
userId(value.login, value.email) : value);
    this.userLogin = userId.login;
    this.userEmail = userId.email;
    this.form = new FormGroup({
      count: new FormControl(1, [Validators.required, Validators.min(1)])
    })
    try {
      // @ts-ignore
      const userId = JSON.parse(localStorage.getItem('user'),
        (key, value) => (typeof value === 'object') && key !== '' ? new
userId(value.login, value.email) : value);
      this.userLogin = userId.login;
```

```
      this.userEmail = userId.email;
      if (this.userLogin.length > 0) this.authorized = true;
    }
    catch (IOEException){}
  }
  get email(){
    return this.form.controls.count as FormControl
  }

  onSubmit(){
    const formData = this.form.value;
    this.productService.addToOrder({user:this.userLogin,
count:formData.count, price:this.product.price, image:this.product.image, ti-
tle:this.product.title, description:this.product.description, cate-
gory:this.product.category}).subscribe()
  }
}
```

focus.directive.ts

```
import {AfterViewInit, Directive, ElementRef, OnInit} from '@angular/core';

@Directive({
  selector: '[appFocus]'
})
export class FocusDirective implements OnInit, AfterViewInit{

  constructor(private el: ElementRef) { }

  ngAfterViewInit(): void {
    this.el.nativeElement.focus()
  }

  ngOnInit(): void {
  }

}
```

account.component.ts

```
import { Component, OnInit } from '@angular/core';
import {AuthService} from "../../services/auth.service";
import {ProductsService} from "../../services/products.service";

@Component({
  selector: 'app-account-page',
  templateUrl: './account-page.component.html',
  styleUrls: ['./account-page.component.scss']
})
export class AccountPageComponent implements OnInit {
  userLogin: string;
  userEmail: string;
  userPassword: string;
  constructor(public productsService: ProductsService) { }
  term = ''
  ngOnInit(): void {
    // @ts-ignore
    const userId =  JSON.parse(localStorage.getItem('user'),
      (key, value) => (typeof value === 'object') && key !== '' ? new
userId(value.login, value.email,value.password) : value);
    this.userLogin = userId.login;
```

```
    this.userEmail = userId.email;
    this.userPassword = userId.password;
    //this.productsService.getFavs(this.userLogin);
    this.productsService.getAll().subscribe(()=>{});
  }

}
```

login-page.component.ts

```
import { Component, OnInit } from '@angular/core';
import {FormControl, FormGroup, Validators} from "@angular/forms";
import {UserService} from "../../services/user.service";
import {User} from "../../models/user";
import {ActivatedRoute, Params, Router} from "@angular/router";
import {AuthService} from "../../services/auth.service";
import {delay} from "rxjs";

@Component({
  selector: 'app-login-page',
  templateUrl: './login-page.component.html',
  styleUrls: ['./login-page.component.scss']
})
export class LoginPageComponent implements OnInit {

  constructor(
    public authService: AuthService,
    private router: Router,
    private route: ActivatedRoute,
    private userService: UserService
  ) { }
  form:FormGroup;
  ngOnInit(): void {
    this.form = new FormGroup({
      email: new FormControl('',[Validators.required, Validators.email]),
      password: new FormControl('',[Validators.required, Valida-
tors.minLength(8)])
    })
  }
  check = true;
  get email(){
    return this.form.controls.email as FormControl
  }
  get password(){
    return this.form.controls.password as FormControl
  }
  onSubmit(){
    const formData = this.form.value;
    this.userService.getUsers(formData.email)
      .subscribe((user:User[]) => {
        if(user[0]){
          if(user[0].password === formData.password){
            this.check = true;
            window.localStorage.setItem('user',JSON.stringify(user[0]));
            this.authService.ilogin(user[0].login);
            this.router.navigate(['/'],{});
          }
          this.check = false;
        }
      })
  }
}
```

order-page.component.ts

```
import {Component, Input, OnInit} from '@angular/core';
import {ProductsService} from "../../services/products.service";
import {IOrder} from "../../models/order";
import {IProduct} from "../../models/product";
import {ModalService} from "../../services/modal.service";
import {delay, isEmpty} from "rxjs";

@Component({
  selector: 'app-order-page',
  templateUrl: './order-page.component.html',
  styleUrls: ['./order-page.component.scss']
})
export class OrderPageComponent implements OnInit {

  title = 'Basket';
  userLogin: string;
  userEmail: string;
  order:IOrder[]
  loading = false
  ording = false
  ordered = false
  empty=true
  term = ''

  constructor(public productsService: ProductsService, public modalService:
ModalService) {
  }

  ngOnInit(): void {
    this.loading = true
    // @ts-ignore
    const userId = JSON.parse(localStorage.getItem('user'),
      (key, value) => (typeof value === 'object') && key !== '' ? new
userId(value.login, value.email) : value);
    this.userLogin = userId.login;
    this.userEmail = userId.email;
    this.productsService.getOrders(this.userLogin).subscribe(o => {
      this.order = o
      if(o.length != 0){
        this.empty = false
      }
    })
  }
  removeFullOrder():void{
    console.log(this.order)
    for(let i in this.order) {
      // @ts-ignore
      this.productsService.removeOrder(this.order[i].id).subscribe(()=>{})
      delay(1)
    }
    location.reload()
  }
  doOrder():void{
    for(let i in this.order) {
      this.productsService.doOrder(this.order[i]).subscribe(()=>{})
    }
    this.removeFullOrder()
  }}
```

product-page.component.ts

```
import { Component, OnInit } from '@angular/core';
import {ProductsService} from "../../services/products.service";
import {ModalService} from "../../services/modal.service";

@Component({
  selector: 'app-product-page',
  templateUrl: './product-page.component.html',
  styleUrls: ['./product-page.component.scss']
})
export class ProductPageComponent implements OnInit {

  title = 'Шмотки';
  userLogin: string;
  userEmail: string;
  authorized = false;


  loading = false



  term = ''

  constructor(public productsService: ProductsService, public modalService:
ModalService) {
  }

  ngOnInit(): void {
    this.loading = true

    this.productsService.getAll().subscribe(() => {
      this.loading = false
    })
    // @ts-ignore
    const userId =  JSON.parse(localStorage.getItem('user'),
      (key, value) => (typeof value === 'object') && key !== '' ? new
userId(value.login, value.email) : value);
    this.userLogin = userId.login;
    this.userEmail = userId.email;

    try {
      // @ts-ignore
      const userId = JSON.parse(localStorage.getItem('user'),
        (key, value) => (typeof value === 'object') && key !== '' ? new
userId(value.login, value.email) : value);
      this.userLogin = userId.login;
      this.userEmail = userId.email;
      if (this.userLogin.length > 0) this.authorized = true;
    }
    catch (IOEException){}

  }

}
```

register-page.component.ts

```
import { Component, OnInit } from '@angular/core';
import {AuthService} from "../../services/auth.service";
import {ActivatedRoute, Router} from "@angular/router";
import {UserService} from "../../services/user.service";
import {FormControl, FormGroup, Validators} from "@angular/forms";
import {User} from "../../models/user";

@Component({
  selector: 'app-register-page',
  templateUrl: './register-page.component.html',
  styleUrls: ['./register-page.component.scss']
})
export class RegisterPageComponent implements OnInit {

  constructor(
    private authService: AuthService,
    private router: Router,
    private route: ActivatedRoute,
    private userService: UserService
  ) { }
  form:FormGroup;
  check = true;
  emailCheck = false;
  loginCheck = false;
  ngOnInit(): void {
    this.form = new FormGroup({
      email: new FormControl('',[Validators.required, Validators.email]),
      login: new FormControl('',[Validators.required, Valida-
tors.minLength(3)]),
      password: new FormControl('',[Validators.required, Valida-
tors.minLength(8)]),
      confPass: new FormControl('')
    })

  }
  get confPass(){
    return this.form.controls.confPass as FormControl
  }
  get email(){
    return this.form.controls.email as FormControl
  }
  get egor(){
    return this.form.controls.egor as FormControl

  }
  get login(){
    return this.form.controls.login as FormControl
  }
  get password(){
    return this.form.controls.password as FormControl
  }
  onSubmit(){
    const formData = this.form.value;
    this.userService.getUsers(formData.email)
      .subscribe((user:User[]) => {
        if(formData.confPass == formData.password) {
          this.check = true;
          console.log("1");
          if (user.length==0) {
            this.emailCheck = false;
```

```
            this.userService.getUsers(formData.login).sub-
scribe((user:User[])=>{
              if(user.length==0){
                this.loginCheck = false;
                this.userService.createUser({
                  email: this.form.value.email as string,
                  password: this.form.value.password as string,
                  login: this.form.value.login as string
                }).subscribe(() => {
                  this.router.navigate(['/login'], {
                    queryParams: {
                      canLogin: true,
                    }
                  });
                });
              }
              else this.loginCheck = true;
            })
          }
          else this.emailCheck = true;
        }
        else this.check=false;
        console.log(formData.password, formData.confPass);
      })
    }
  }
```

filter-product.pipe.ts

```
import { Pipe, PipeTransform } from '@angular/core';
import {IProduct} from "../models/product";

@Pipe({
  name: 'filterProducts'
})
export class FilterProductsPipe implements PipeTransform {

  transform(products: IProduct[] = [], search: string = ''): IProduct[] {
    if(search.length === 0) return products
    return products.filter(product => product.title.toLowerCase().in-
cludes(search.toLowerCase())) ||
      products.filter(product => product.price === Number(search)) ||
      products.filter(product => product.category.toLowerCase().in-
cludes(search.toLowerCase())) ||
      products.filter(product => product.description.toLowerCase().in-
cludes(search.toLowerCase()));
  }
}
```

auth.service.ts

```
import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root'
})
export class AuthService {

  constructor() { }
  public isAuthentificated: boolean = false;
```

```
  public login:string;

  ilogin(login:string): void{
    console.log(this.login);
    this.isAuthentificated = true;
    this.login = login;
    location.replace("/");
  }
  ilogout(): void{
    this.isAuthentificated = false;
    window.localStorage.clear();
  }
}
```

error.service.ts

```
import { Injectable } from '@angular/core';
import {Subject} from "rxjs";

@Injectable({
  providedIn: 'root'
})
export class ErrorService {

  error$ = new Subject<string>()

  handle(message: string){
    this.error$.next(message)
  }

  clear(){
    this.error$.next('')
  }
}
```

modal.service.ts

```
import { Injectable } from '@angular/core';
import {BehaviorSubject} from "rxjs";

@Injectable({
  providedIn: 'root'
})
export class ModalService {

  isVisible$ = new BehaviorSubject<boolean>(false)
  open(){
    this.isVisible$.next(true)
  }
  close(){
    this.isVisible$.next(false)
  }
}
```

product.service.ts

```
import {Injectable} from "@angular/core";
import {HttpClient, HttpErrorResponse, HttpParams} from "@angular/com-
mon/http";
import {catchError, Observable, retry, tap, throwError} from "rxjs";
```

```
import {IProduct} from "../models/product";
import {ErrorService} from "./error.service";
import {products} from "../data/products";
import {IOrder} from "../models/order";

@Injectable({
  providedIn: 'root'
})
export class ProductsService {
  constructor(
    private http: HttpClient,
    private errorService: ErrorService
  ) {}

  products: IProduct[] = []
  orders:IOrder[] = []

  getAll(): Observable<IProduct[]> {// запрос на сервак и получение данных
    return this.http.get<IProduct[]>('http://localhost:3000/Products', {
      params: new HttpParams({
        fromObject: {limit: 10}
      })
    }).pipe(
      //delay(2000),
      retry(5),
      tap(products => this.products = products),
      catchError(this.errorHandler.bind(this))
    )
  }

  create(product: IProduct): Observable<IProduct>{
    return this.http.post<IProduct>('http://localhost:3000/Products', prod-
uct)
      .pipe(
        tap(prod => this.products.push(prod))
      )
  }
  addToOrder(order: IOrder) : Observable<IOrder>{
    return this.http.post<IOrder>(`http://localhost:3000/Orders`, or-
der).pipe(
      tap(ord => this.orders.push(ord))
    )
  }
  getOrders(user: string): Observable<IOrder[]>{
    return this.http.get<IOrder[]>(`http://localhost:3000/Or-
ders?user=${user}`).pipe(
      tap(ord => this.orders = ord)
    )
  }
  removeOrder(id:number): Observable<IOrder>{
    return this.http.delete<IOrder>(`http://localhost:3000/Orders/${id}`);
  }
  doOrder(order:IOrder):Observable<IOrder>{
    return this.http.post<IOrder>(`http://localhost:3000/Ordered`, order);
  }
  private errorHandler(error: HttpErrorResponse){
    this.errorService.handle(error.message)
    return throwError(() => error.message)
  }
}
```

user.service.ts

```
import {Injectable} from '@angular/core';
import {HttpClient} from '@angular/common/http';
import {Observable, tap} from 'rxjs';
import {User} from '../models/user';

@Injectable({
  providedIn: 'root'
})
export class UserService{
  constructor(private http: HttpClient) {
  }
  users:User[] = [];
  getUsers(email: string): Observable<User[]> {
    return this.http.get<User[]>(`http://localhost:3000/us-
ers?email=${email}`)
      .pipe(tap(users => this.users = users));
  }
  createUser(user: User) : Observable<User>{
    return this.http.post<User>('http://localhost:3000/users', user).pipe(
      tap(user => this.users.push(user))
    )
  }
}
```

app.module.ts

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import {ProductComponent} from "./components/product/product.component";
import {HttpClientModule} from "@angular/common/http";
import { GlobalErrorComponent } from './components/global-error/global-er-
ror.component';
import {FormsModule, ReactiveFormsModule} from "@angular/forms";
import { FilterProductsPipe } from './pipes/filter-products.pipe';
import { ModalComponent } from './components/modal/modal.component';
import { CreateProductComponent } from './components/create-product/create-
product.component';
import { FocusDirective } from './directives/focus.directive';
import { ProductPageComponent } from './pages/product-page/product-page.com-
ponent';
import { AboutPageComponent } from './pages/about-page/about-page.component';
import { NavigationComponent } from './components/navigation/navigation.com-
ponent';
import {UserService} from './services/user.service';
import { LoginPageComponent } from './pages/login-page/login-page.component';
import { RegisterPageComponent } from './pages/register-page/register-
page.component';
import { AccountPageComponent } from './pages/account-page/account-page.com-
ponent';
import { OrderComponent } from './components/order/order.component';
import { OrderPageComponent } from './pages/order-page/order-page.component';
@NgModule({
  declarations: [
    AppComponent,
    ProductComponent,
```

```
    GlobalErrorComponent,
    FilterProductsPipe,
    ModalComponent,
    CreateProductComponent,
    FocusDirective,
    ProductPageComponent,
    AboutPageComponent,
    NavigationComponent,
    LoginPageComponent,
    RegisterPageComponent,
    AccountPageComponent,
    OrderComponent,
    OrderPageComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    HttpClientModule,
    FormsModule,
    ReactiveFormsModule
  ],
  providers: [UserService],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

app-routing.module.ts

```
mport { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import {ProductPageComponent} from "./pages/product-page/product-page.compo-
nent";
import {AboutPageComponent} from "./pages/about-page/about-page.component";
import {LoginPageComponent} from "./pages/login-page/login-page.component";
import {RegisterPageComponent} from "./pages/register-page/register-page.com-
ponent";
import {AccountPageComponent} from "./pages/account-page/account-page.compo-
nent";
import {OrderPageComponent} from "./pages/order-page/order-page.component";

const routes: Routes = [
  {path: '', component: ProductPageComponent},
  {path: 'about', component: AboutPageComponent},
  {path: 'login', component: LoginPageComponent},
  {path: 'register', component: RegisterPageComponent},
  {path: 'account', component: AccountPageComponent},
  {path: 'orders', component: OrderPageComponent}
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```