

Functionalitate Proiect:

In acest document voi mentiona pas cu pas fiecare cerinta implementata si unde anume in cod, pentru o cautare mai rapida dar si pentru a intelege proiectul.

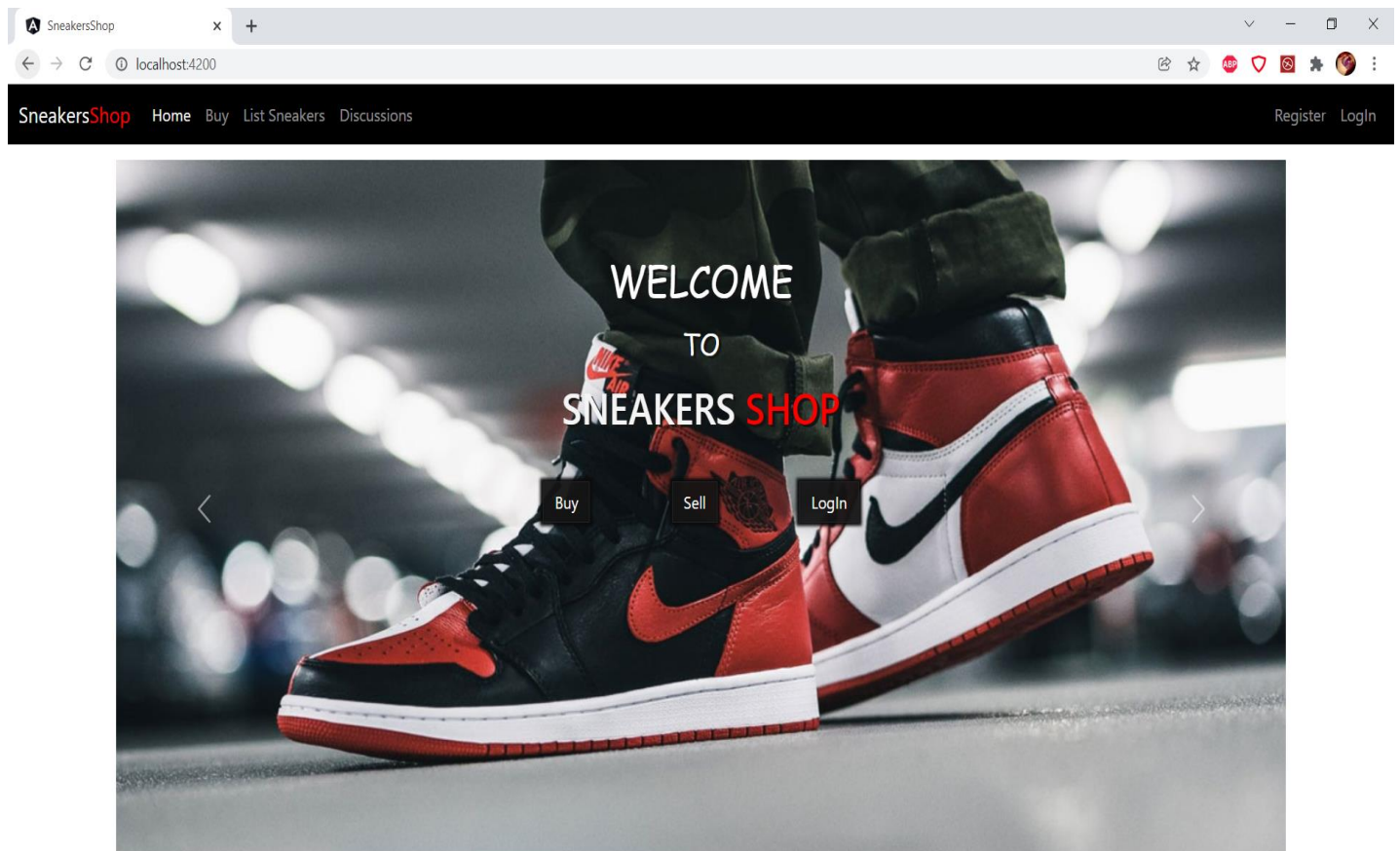
Tematica:

Proiectul realizeaza cu succes implementarea front-end a unui site asemanator cele de tipul E-commerce pentru incaltaminte, dar permitant fiecarui utilizator inregistrat sa isi promoveze si vanda propriile produse, astfel fiind un site dedicat pentru Resell.

Utilizare:

In momentul in care utilizatorul acceseaza site-ul, acesta este intampinat de mesajul "Welcome" aflat pe prima pagina (home-page). De aici are acces atat la butuanele din mijlocul ecranului, cat si la bara de navigare, putand sa navigheze cu usurinta intre componentele implementate.

De asemenea, are implementat si un carusel cu tematica corespunzatoare.



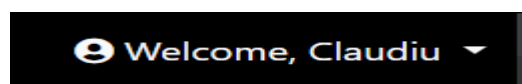
Pentru a accesa unele pagini precum "List Sneakers" sau "Discussions" este nevoie ca utilizatorul sa se logheze / inregistreze apasand pe butonul de "Register" in cazul in care nu este deja redirectionat automat. Intrucat proiectul prevede doar implementarea front-end, neavand o baza de date, datele salvate din formularul de inregistrare sunt salvate in Local Storage.

The screenshot shows a web browser window with the address bar displaying 'localhost:4200/user-register'. The page has a dark header with the 'SneakersShop' logo and navigation links: Home, Buy, List Sneakers, Discussions. On the right side of the header are links for 'Register' and 'Login'. The main content area features a 'Register' form with a dark header. The form contains five input fields: Name, Email, Password, Confirm Password, and Mobile. At the bottom of the form are two buttons: 'Register' and 'Cancel'.

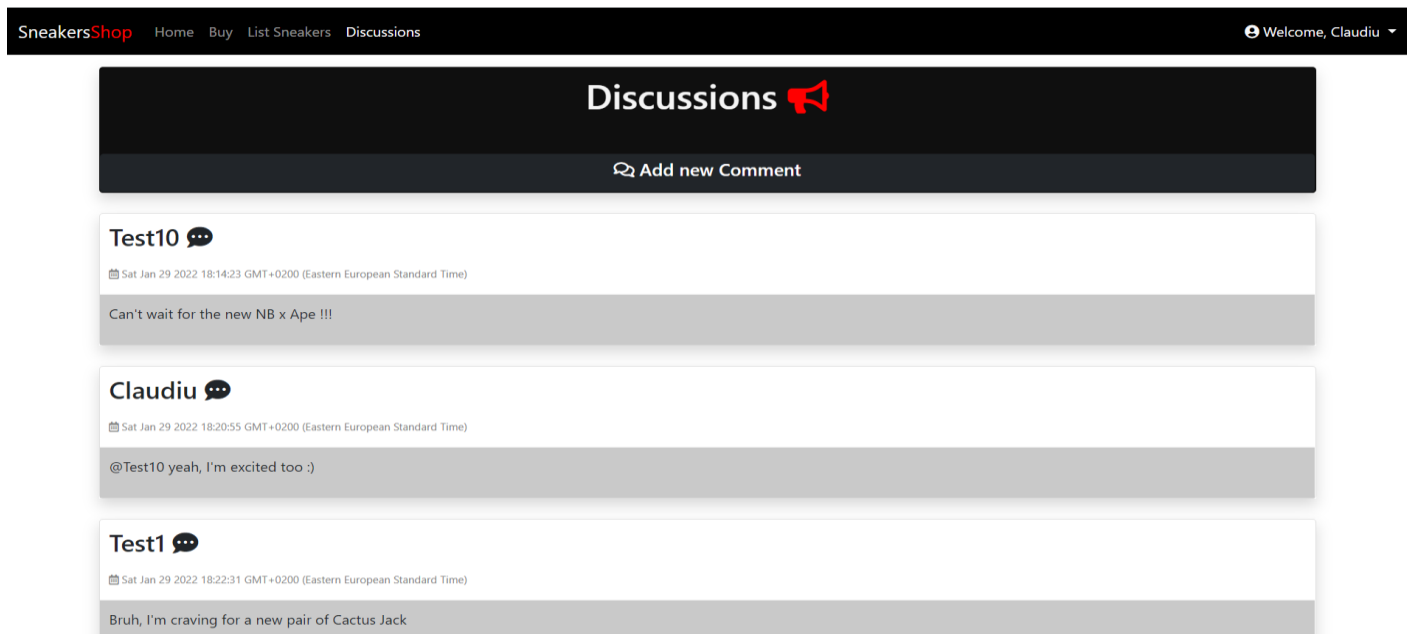
Odata inregistrat, utilizatorul este redirectionat spre formularul de Login.

The screenshot shows the 'Login' form, which is a modal or a page with a dark header. The form has two input fields: Name and Password. Below the password field is a link that says 'Don't have an account yet?'. At the bottom of the form are two buttons: 'Login' and 'Cancel'.

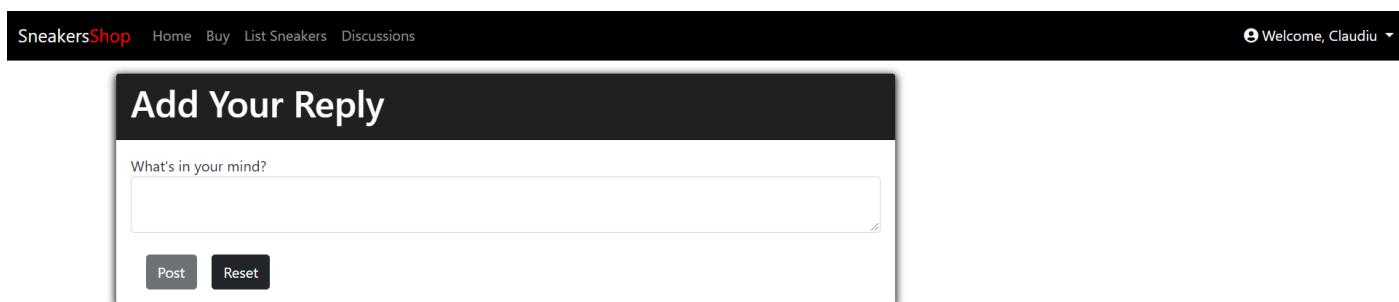
In urma logarii, se creeaza un 'token' in Local Storage care tine minte utilizatorul curent.



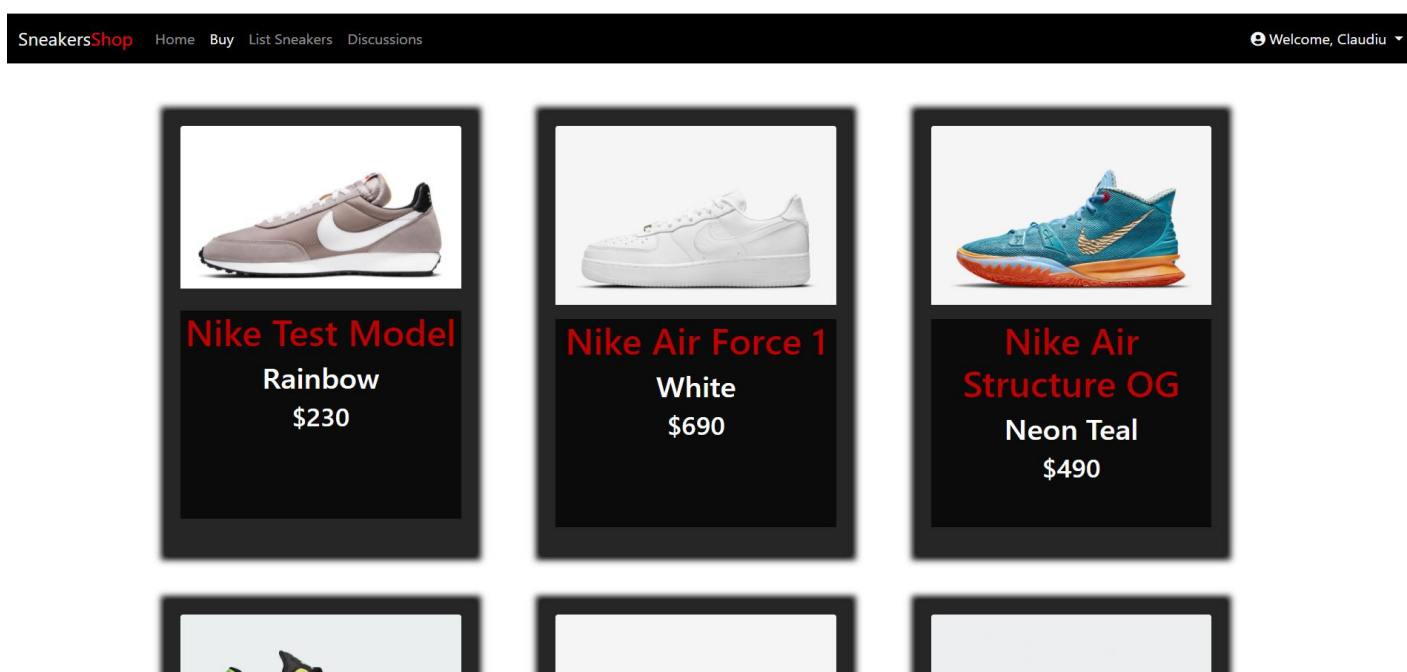
Acum Utilizatorul poate naviga oriunde pe site-ul nostru fara a mai fi oprit / redirectionat. Acesta poate vizita produsele listate in pagina 'Buy' sau poate lista chiar el noi produse in 'List Sneakers'. De asemenea, poate interactiona si cu alti utilizatori in sectiunea 'Discussions':



Poate adauga un nou comentariu apasand pe butonul 'Add new Comment' si completand formularul corespunzator:




In pagina 'Buy-Page' utilizatorul poate sa cumpere si sa acceseze un anumit produs plasand mouse-ul deasupra imaginii si apasand pe cosul de cumparaturi din stanga.





Preview detaliat produs:

SneakersShop [Home](#) [Buy](#) [List Sneakers](#) [Discussions](#) Welcome, Claudiu



Nike Air Structure OG
Color: Neon Teal
Price: \$490

Buy  Save 

Details

Brand:
Nike

Model:
Garcia

Release Date:
2020-10-12

In pagina 'List-Sneaker' , utilizatorul are un preview live asupra cartonasilui ce va fii adaugat in Shop. Orice camp din formular modifica se va reflecta pe modelul alaturat din dreapta.

SneakersShop [Home](#) [Buy](#) [List Sneakers](#) [Discussions](#) Welcome, Claudiu

List Your Sneakers

Display Info Details


Name

Color

Price

Next Back

Card List Preview



Rezolvare Cerinte:

Salut, va trimit inca o data cerintele, nu s-a schimbat nimic dar am pus si punctaju la fiecare cerinta. Reamintesc ca nota este 8p proiectul, 2p prezenta si 1p Bonus pentru functionalitati extra.

1. Aceasta este lista finala de cerinte pt Frontend (4p)

Orice numar se specifica in cerinte se inmulteste cu numarul de coechipieri

Ex. Cerinta 1 cere cel putin 3 componente. Daca sunteti 2 in echipa => 6 componente. Daca sunteti 3 => 9 componente.

- Cel putin 3 componente. Existenta rutelor(simple + parametru). (0.5 p)
- Cel putin 3 servicii conectate la serverul din .Net . Afisarea de date din servicii in componente. (1p)
- Cel putin 2 formulare. (Reactive forms) - inafara de login/register (0.5 p)
- Implementarea metodelor de comunicare intre componente. (neaparat sa fie folosita macar 1 data comunicarea printr-un serviciu) (0.5p)
- Cel putin 1 directiva. (pe langa cea facuta la laborator) (0.25)
- Cel putin 1 pipe 0.25
- Register + Login + Implementare Guard (1p)

2. Backend (4p) :

Aceeasi regula, orice numar se specifica in cerinte se inmulteste cu numarul de coechipieri

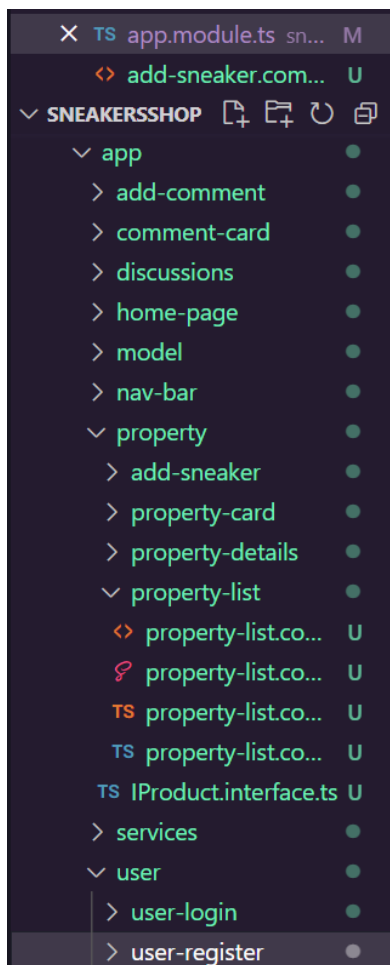
- 3 Controllere (minim); Fiecare Metoda Crud, REST cu date din baza de date. (1p)
- Cel puțin 1 relație între tabele din fiecare fel (One to One, Many to Many, One to Many); Folosirea metodelor din Linq: GroupBy, Where, etc; Folosirea Join si Include (1p)
- Autentificare + Roluri; Autorizare pe endpointuri în funcție de Roluri; Cel puțin 2 Roluri: Admin, User (1p)
- Sa se foloseasca repository pattern/unit of work (1p)

3. Prezente (2p)

Am avut 9 laboratoare in total care au contat la prezenta deci 1 prezenta valoreaza 0.23 p

1) Cel putin 3 Componente + Rute (simple + parametru) :

- Avem implementat un numar consistent de componente (**11 componente** : *add-comment, comment-card, discussions, home-page, nav-bar, add-sneaker, property-card, property-details, property-list, user-login, user-register*)



Toate aceste componente sunt importate si in app.modules.ts :

```

11 import { AppRoutingModuleModule } from './app-routing.module';
12 import { AppComponent } from './app.component';
13 import { PropertyCardComponent } from './property/property-card/property-card.component';
14 import { PropertyListComponent } from './property/property-list/property-list.component';
15 import { NavBarComponent } from './nav-bar/nav-bar.component';
16 import { AddSneakerComponent } from './property/add-sneaker/add-sneaker.component';
17 import { HomeComponent } from './home-page/home-page.component';
18 import { NgbModule } from '@ng-bootstrap/ng-bootstrap';
19 import { PropertyDetailsComponent } from './property/property-details/property-details.component';
20 import { SnkServiceService } from './services/snk-service.service';
21 import { UserLoginComponent } from './user/user-login/user-login.component';
22 import { UserRegisterComponent } from './user/user-register/user-register.component';
23 import { UserServiceService } from './services/user-service.service';
24 import { AlertServiceService } from './services/alert-service.service';
25 import { AuthServiceService } from './services/auth-service.service';
26 import { AuthGuardGuard } from './services/auth-guard.guard';
27 import { MiddlemanService } from './services/middleman.service';
28 import { CurrencyPipe } from '@angular/common';
29 import { CheckAuthGuard } from './services/check-auth.guard';
30 import { LiftLogoDirective } from './lift-logo.directive';
31 import { LogoFontDirective } from './logo-font.directive';
32 import { RainbowInputDirective } from './rainbow-input.directive';
33 import { CommentCardComponent } from './comment-card/comment-card.component';
34 import { AddCommentComponent } from './add-comment/add-comment.component';
35 import { DiscussionsComponent } from './discussions/discussions.component';

```

- **Route / Routes simple + parametru:**

```

const appRoutes: Routes = [
  {path: '', component: HomeComponent},
  {path: 'buy-sneakers', component: PropertyListComponent},
  {path: 'add-sneakers', component: AddSneakerComponent,
    canActivate: [AuthGuardGuard]},
  {path: 'product/:id', component: PropertyDetailsComponent},
  {path: 'user-login', component: UserLoginComponent,
    canActivate: [CheckAuthGuard]},
  {path: 'user-register', component: UserRegisterComponent},
  {path: 'add-comment', component: AddCommentComponent,
    canActivate: [AuthGuardGuard]},
  {path: 'discussions', component: DiscussionsComponent}
]

```

Toate rutele din

appmodule

- **Folosire route simple: screenshot din nav-bar component**

```

<li class="nav-item">
  <a class="nav-link" [routerLinkActiveOptions]="{exact:true}" routerLinkActive="active" routerLink="/discussions">
</li>

```

- **Folosire route cu parametru: 2 ss-uri din property-card component si implementarea in property-details component.**

```

<li class="list-inline-item">
  <button routerLink="/product/{property.Id}" class="btn btn-dark">
</li>

```



```

ngOnInit(): void {
  this.productId = Number(this.route.snapshot.params['id'])

  // acelasi lucru ca mai sus, dar pt routerLink:
  this.route.params.subscribe(
    (params) => {
      this.productId = Number(params['id']);
      this.getSnk.getProduct(this.productId).subscribe(
        data => {
          this.product.Name = data!.Name;
          this.product.Type = data!.Type;
          this.product.Price = data!.Price;
          this.product.Image = data!.Image;
          this.product.Brand = data!.Brand;
          this.product.Model = data!.Model;
          this.product.Date = data!.Date;
          this.product.BuyLink = data!.BuyLink;
        }
      )
    }
  )
}

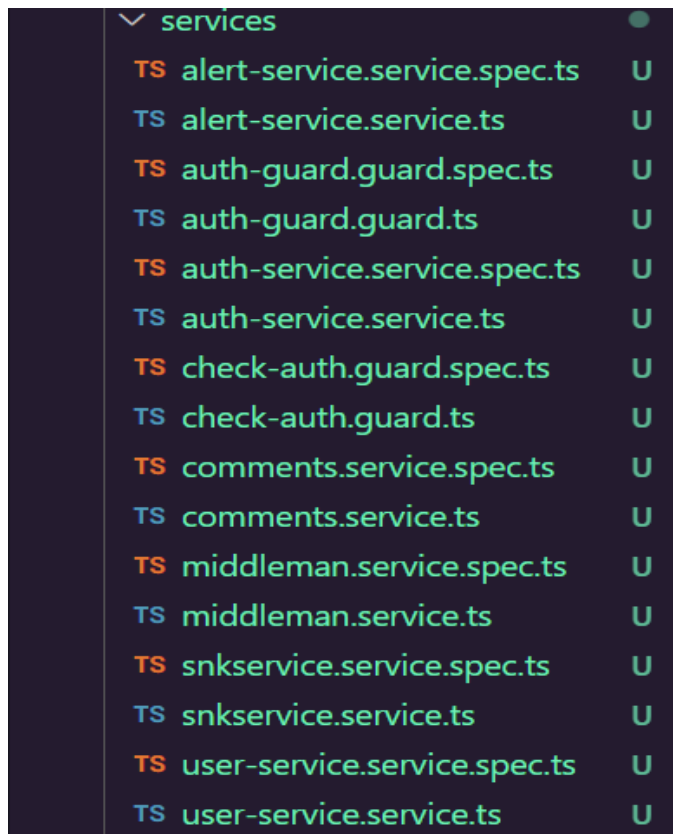
```

Scurta explicatie: Folosim rute parametrizate pentru a afisa informatiile produsului in functie de ID ul sau (eg: Un utilizator adauga al n-lea produs. Folosind rutele parametrizate, cand apasam pe un cartonas cu produs, vom fii redirectionati pe pagina cu detalii generate pentru produsul cu Id-ul 'n')

2) Cel puțin 3 servicii. Afisarea de date din servicii in componente:

!Observatie: Intrucat doar partea de front_end este implementata, serviciile nu sunt conectate la serverul din .Net (acesta fiind inexistent).In schimb, serviciile sunt conectate pe partea de **HttpClient**. Ele comunica cu localStorage, cu componentele aplicatiei si intre ele si unele folosesc metode de 'get' si 'set' pentru preluarea si/sau modificarea key-lor si valorilor din localStorage sau din componente.

- **Servicii (Services)** : Avem 6 servicii clasice + 2 servicii tip Guard:



- **Afisarea de informatii / mesaje / date din servicii in componente:** ne vom folosi de anumite servicii fie pentru a popula cartonasel de produse din componenta *property-list* si *property-card* cu informatiile salvate in localStorage, fie pentru a afisa o notificare intr-o componenta. (eg: O componenta va apela serviciul (folosind si router) care va afisa mesajul in alta pagina / componenta sub forma de pop-up notification.

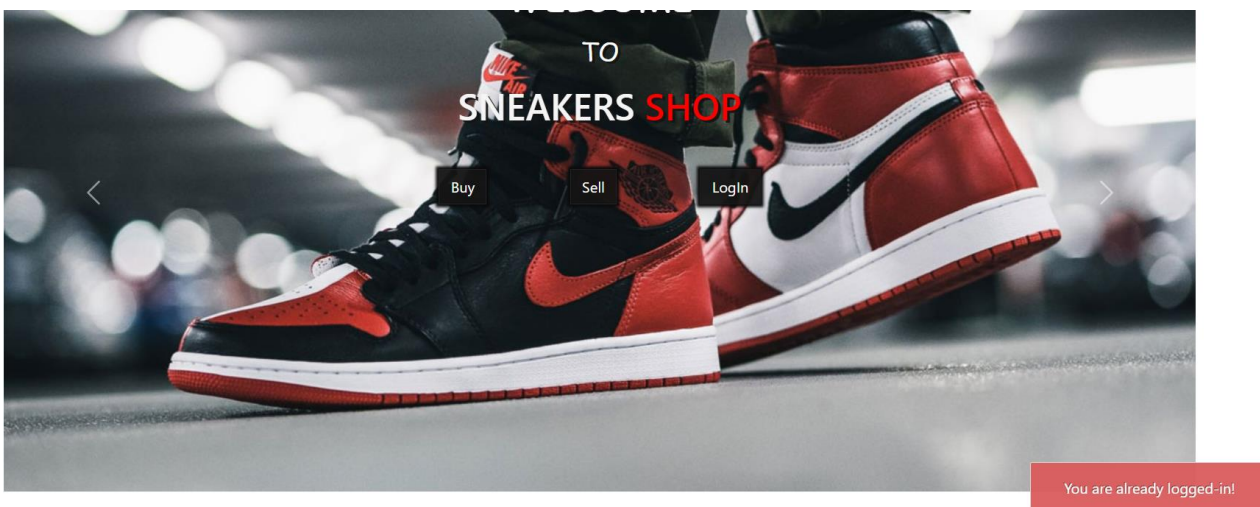
```
Properties!: IProduct[];  
  
// Servicii. Afisarea / maparea de date din Local storage in componenta prin servicii:  
constructor(private snkService: SnkServiceService) { }  
  
ngOnInit(): void {  
  this.snkService.getAllSneakers().subscribe(  
    data=>{  
      this.Properties=data;  
    }  
  )  
}
```

Aici folosim serviciul snkService ce are functia getAllSneakers() pentru a completa cartonasel cu produse din *property-list* component cu informatii.


```
export class CheckAuthGuard implements CanActivate {
  constructor(private userAlert: AlertServiceService,
               private middleman: MiddlemanService,
               private router: Router) { }

  canActivate() {
    if (this.middleman.currentStatus()) {
      this.router.navigate(['/']);
      this.userAlert.errorf("You are already logged-in!");
      return false;
    } else {
      return true;
    }
  }
}
```

Aici , in interiorul serviciului tip Guard denumit check-auth folosindu-ne de serviciul 'AlertService' vom afisa o notificare in componenta *home-page* daca utilizatorul este deja conectat, asa cum se vede mai jos:



- **Folosirea metodelor 'get' si 'set'**, dar pe parte de HttpClient importata in angular, manipuland date din localStorage (intrucat avem doar front-end): ss *din* 'snk-service'

```
constructor(private http:HttpClient) { }

getAllSneakers(): Observable<IProductDets[]>{
  // Implementare folosind pipe: Transformam raspunsul HTTP cu PIPE pentru tipul definit in interfata
  return this.http.get('data/properties.json').pipe(
    map( data =>{
      if(localStorage.getItem('newSnk'))
      {
        newProd = [...JSON.parse(localStorage.getItem('newSnk') as string), product]
      }
      localStorage.setItem('newSnk', JSON.stringify(newProd));
    })
  )
}
```

3) Cel puțin 2 formulare Reactive forms pe lângă login/register:

Avem în total 4 formulare: 2 formulare pentru login și register și încă 2 formulare pentru a lista noi produse și pentru a adăuga comentarii în discuție.

- **Formular Reactive** – adăuga o nouă componentă: ss din componenta add-sneaker

The screenshot shows the SneakersShop application interface. At the top is a navigation bar with links: SneakersShop, Home, Buy, List Sneakers, Discussions, and a user profile 'Welcome, Claudiu'. Below the navigation bar, there are two main sections. On the left is the 'List Your Sneakers' form, which has two tabs: 'Display Info' (selected) and 'Details'. The form contains input fields for 'Name', 'Color', and 'Price', and 'Next' and 'Back' buttons at the bottom. On the right is the 'Card List Preview' section, which displays a card for a sneaker with the text '404 ERROR' and a placeholder image.

```
<div class= "card-body">
  <form [formGroup]="listingForm" (ngSubmit)="onSub()">
    <div>
      <tabset #staticTabs>
```

- **Formular Reactive** – adăugarea unui comentariu:

The screenshot shows the SneakersShop application interface. At the top is a navigation bar with links: SneakersShop, Home, Buy, List Sneakers, Discussions, and a user profile 'Welcome, Claudiu'. Below the navigation bar, there is a section titled 'Add Your Reply'. It contains a text input field with the placeholder text 'What's in your mind?' and 'Post' and 'Reset' buttons at the bottom.

```
<div class= "card-body">
  <form [formGroup]="commentForm" (ngSubmit)="onSubComm()">
    <div class = "form-group col-12">
```

4) Metoda de comunicare între componente (comunicare printr-un serviciu):

Am implementat metoda de comunicare între componente printr-un serviciu intitulat **"MiddlemanService"**. Acesta conține o variabilă tip `bool` și o funcție ce verifică și schimbă această variabilă (dacă utilizatorul este conectat -> variabila este `true` iar funcția va returna `true`, iar dacă utilizatorul nu este conectat -> funcția întoarce `false`). Statusul serviciului este setat la `true` în componenta `'user-login'` (atunci când utilizatorul se loghează). Astfel, dacă utilizatorul navighează între pagini / componente, acestea verifică statusul returnat de serviciul `"middleman"` (adică verifică dacă utilizatorul este conectat sau nu). Dacă utilizatorul apasă `"LogOut"` în oricare pagină/componentă s-ar afla, componenta respectivă va apela serviciul `"middleman"` și va schimba valoarea variabilei `bool` în `false`, iar atunci când utilizatorul va naviga spre altă pagină, statusul întors în acea componentă va fi `false`. Astfel, componentele comunică prin acest serviciu.

```

export class MiddlemanService {

  isLoggedIn!: boolean;
  loggedUser!: string;
  constructor() { }

  checkLog()
  {
    this.loggedUser = localStorage.getItem('token') as string;
    if(this.loggedUser)
    {
      this.isLoggedIn=true;
    }
    else
    {
      this.isLoggedIn=false;
    }
  }

  currentStatus()
  {
    return this.isLoggedIn;
  }
}

```

5) Cel puțin o directiva:

Am folosit **3 directive**: Avem o directiva care modifica pozitia titlului “Welcome” din componenta ‘home-page’, o directiva care seteaza alt font pentru tagul <h1> si o directiva ce foloseste HostListener si HostBinding pentru a crea un input field rainbow (folosita in formularul de adaugat produs).

TS lift-logo.directive.spec.ts	U
TS lift-logo.directive.ts	U
TS logo-font.directive.spec.ts	U
TS logo-font.directive.ts	U
TS rainbow-input.directive.spec.ts	U
TS rainbow-input.directive.ts	U

- Screenshot directiva *rainbow-input*:

```

@Directive({
  selector: '[rainbowField]'
})
export class RainbowInputDirective {

  possibleColors = [
    'lime',
    'indigo',
    'blue',
    'orange',
    'yellow',
    'green'
  ];

  @HostBinding('style.color') color!: string;

  @HostListener('keydown') newColor() {
    const colorPick = Math.floor(Math.random() * this.possibleColors.length);
    this.color = this.possibleColors[colorPick];
  }
}

```

6) Cel puțin un pipe:

Am folosit un "CurrencyPipe" în componenta "add-sneaker" pentru a seta automat unitatea monetară ca fiind \$ pentru câmpul price al formularului de adăugare produs:

De asemenea, am folosit și metoda pipe() în anumite servicii.

```
import { CurrencyPipe } from '@angular/common';

this.listingForm.valueChanges.subscribe( form=> {
  if(form.snkPrice)
  {
    this.listingForm.patchValue({
      snkPrice: this.currencyPipe.transform(form.snkPrice.replace(/\D/g, '').replace(/^0+/, ''), 'USD',
    }, {emitEvent: false})
    }
  });
```

7) Login + Register + Implementare Guard:

Pe lângă formularele de login și register avem 2 servicii GUARD de tipul canActivate:

```
const appRoutes: Routes = [
  {path: '', component: HomeComponent},
  {path: 'buy-sneakers', component: PropertyListComponent},
  {path: 'add-sneakers', component: AddSneakerComponent,
    canActivate: [AuthGuardGuard]},
  {path: 'product/:id', component: PropertyDetailsComponent},
  {path: 'user-login', component: UserLoginComponent,
    canActivate: [CheckAuthGuard]},
  {path: 'user-register', component: UserRegisterComponent},
  {path: 'add-comment', component: AddCommentComponent,
    canActivate: [AuthGuardGuard]},
  {path: 'discussions', component: DiscussionsComponent}
]
```

The screenshot shows a web browser window with the URL `localhost:4200/user-register`. The page title is "SneakersShop" and the navigation bar includes links for "Home", "Buy", "List Sneakers", and "Discussions". On the right side of the navigation bar, there are links for "Register" and "Login". The main content area displays a "Register" form with the following fields: Name, Email, Password, Confirm Password, and Mobile. At the bottom of the form, there are two buttons: "Register" and "Cancel".

Login

Name

Password

[Don't have an account yet?](#)

Login

Cancel