UpToSale Store App

Up-To-Sale prodavnica

Angular web aplikaciija prodavnice kostruisana od temelja,

pružajući iskustvo sa tipičnim procesom razvoja, kao i uvod u osnovne koncepte,

alate i terminologiju dizajna.

Autor: Djordje Taskovic

Datum: 30.6.2022

Uvod

Aplikacija UpToSale Store pomaze kranjem kosiniku manipulaciju proizvoda na stranici prodavnice. Aplikacija ima mnoge funkcije koje biste očekivali da pronađete u bilo kojoj aplikaciji zasnovanoj na podacima.

Gotova aplikacija preuzima i prikazuje listu proizvoda, uređuje detalje izabranog proizvoda i kreće se između različitih prikaza podataka stanice.

Aplikacija poseduje sledece sposobnosti:

- Koristit ugrađene Angular direktive da bi prikazali i sakrili elemente i prikazali liste podataka
- Kreira Angular komponente da biste prikazali detalje proizvoda i prikazali niz istih
- Koristite jednosmerno povezivanje podataka za podatke samo za čitanje
- Dodata polja koja se mogu uređivati da biste ažurirali model sa dvosmernim povezivanjem podataka
- Povezane metode komponenti sa korisničkim događajima, poput pritiska na tastere i klikova
- Mogucnost korisnika da izaberu proizvod ili karegoriju sa glavne liste i uređuju tog proizvoda u prikazu detalja
- Koriscenje rutiranja za navigaciju između različitih prikaza i njihovih komponenti
- Autentifikacija korisnika pomocu jwt tokena
- Autorizacija i zabrana odredjenih strana ne-aoutrizovanim korisnicima

Karakteristike aplikacije:

- Paginacija sadrzaja
- Autorizacija na osnovu jwt tokena sa servera
- Filter prikaza porizvoda po odabranoj categoriji
- Verifikacija korisninka i formulara nad operacijama sajta
- Vecina CRUD operacija nad elementima aplikacije
- I vise..

Resursi

Api server

Za potrabe aplikacije koriscen je online fake api servis za prokaz podataka.

Platzi Fake Store API se može koristiti sa bilo kojim projektom koji zahteva proizvode, korisnike, kategorije, autentifikaciju i korisnike u JSON formatu. Možete da koristite ovaj API za izradu prototipa e-trgovine i učenje o tome kako da se povežete sa API-jem sa najboljim praksama.

Inicijana url adresa api servsa: Platzi Fake Store API

Karakteristike api servisa sa odgovarajucim endpointovima i konstrukcije data podataka za slanje kroz zadate url adrese, su u vecini objasnjeni na stranici sa detaljnim primerima.

Uvod i konstrukcija endpointova api servisa: Introduction - Platzi Fake Store API

Za potrebe aplikacije korisceni si sledeci end pointovi:

Pristup listi od 200 proizvoda pomoću krajnje tačke /products.

- [GET] https://api.escuelajs.co/api/v1/products

Dohvatenje jednog od proizoda setovanjem id kao parametra

- [GET] https://api.escuelajs.co/api/v1/products/1

Kreiranje novog proizvoda putem slanja objekta sa parametrima

- [POST] https://api.escuelajs.co/api/v1/products/

Obnova postojeceg proizvoda slanjem objekta i dodavanje id parametra

- [PUT] https://api.escuelajs.co/api/v1/products/1

Brisanje proizvoda na osnovu id parametra

- [DELETE] https://api.escuelajs.co/api/v1/products/

Pristup listi od 5 categorija koristeci /categories endpoint

- [GET] https://api.escuelajs.co/api/v1/categories

Dohvatanje jdne categorije setovanjem id parametra

- [GET] https://api.escuelajs.co/api/v1/categories/1

Kreiranje nove kategorije slanjem object-a

- [POST] https://api.escuelajs.co/api/v1/categories

Obnova kategorija na osnovu id paramentra i objekt-a

- [PUT] https://api.escuelajs.co/api/v1/categories/1

Dohvatanje porizvada na osnovi id parametra kategorije

- [GET] https://api.escuelajs.co/api/v1/categories/1/products

Autentifikacija korisnika putem vec zadatih parametra i dohvatanje token objekta za aplikaciju

- [POST] https://api.escuelajs.co/api/v1/auth/login

Koriscene Okruzenja i Tehnologije

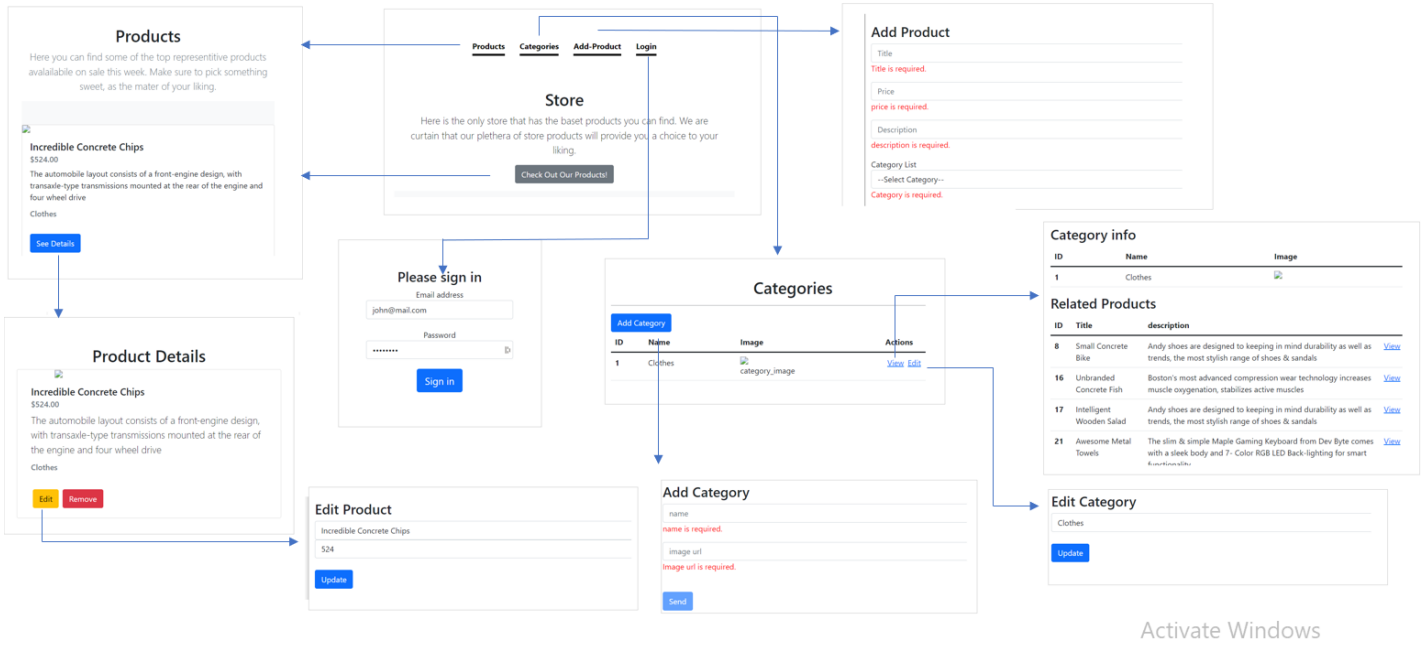
- Visual Studio Code - pisanje i editovanje koda aplikacije
- TypeScript – koriscenje typescript jezika u aplikaciji
- Angular CLI version .14 – koriscenje cli komandi za kreiranje elemenata aplikacija
- Microsoft Edge Browser – Prokaz i testiranje aplikacije
- Angular Rxjs biblioteka – funcionalnosti za prikaz i kreiranje novih obesvable instanci

Korisceni template

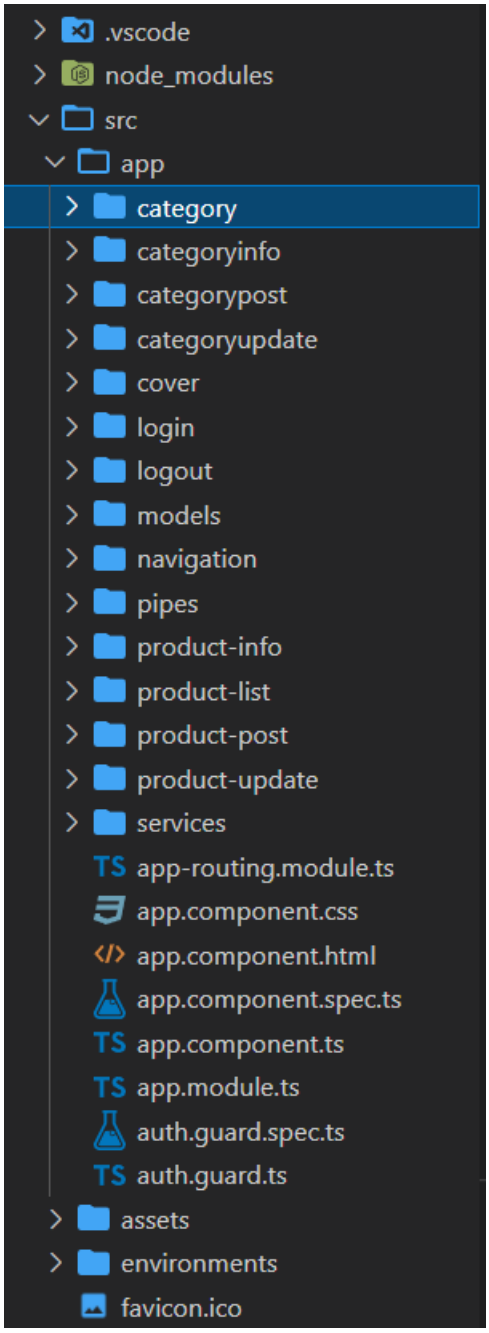
Za potrebe aplikacije iskoriscena je Bootstrap 4 template sa uvezenim stilovima i ogranicenjima za strukturalni prikaz elemenata.

Dijagrami organizacije

Dijagram stranica aplikacije



Snimak projekta aplikacije



Opis fungcionalnosti elemenata projekta

category.component

Dohvatanje podataka sa categoryServica i prikaz istih podataka na stanizi u tabelarnom prikazu.

```
export class CategoryComponent implements OnInit {
  public data!:Observable<Category[]>;

    constructor(private catService: CategoriesService,) {

        this.data = this.catService.getCategories();
      }

  ngOnInit(): void {
  }

}
```

```
  <tr *ngFor="let d of data|async">
            <th scope="row">{{ d.id }}</th>
            <td>{{ d.name }}</td>
            <td> <img class="d-block w-25" src="{{ d.image }}"
alt="category_image" /></td>
            <td>
                <a class="m-1" [routerLink]="['/categoryinfo',d.id]">View</a>
                <a class="m-1"
[routerLink]="['/editcategories',d.id]">Edit</a>
            </td>
        </tr>
```

Categoryinfo.component

Ispisavanje zasebne kategorije na osnovu id parametra iz izdvojenog iz http string zahteva,

Prikaz povezani proizvoda za odabranu kategoriju.

```
export class CategoryinfoComponent implements OnInit {
  public id!:string|null;
  public category!:Category;
  public products!:Observable<InTwineProduct[]>;


  constructor(private route:ActivatedRoute,
    private catService:CategoriesService) {

    this.route.paramMap.subscribe( par => {
```

```
      this.id = par.get('id');
    });

    this.catService.getCategoryById(this.id)
    .subscribe((e:Category) => {
      this.category = e;
      });

   this.products = this.catService.getProductsbyCategory(this.id);
  }

  ngOnInit(): void {
  }

}
```

```
<h2>Related Products</h2>
<table class="table">
    <thead>
        <tr>
            <th scope="col">ID</th>
            <th scope="col">Title</th>
            <th scope="col">description</th>
        </tr>
    </thead>
    <tbody>
        <tr *ngFor="let product of products|async">
            <th scope="row">{{ product.id }}</th>
            <td>{{ product.title }}</td>
            <td> {{ product.description }}</td>
            <td><a class="m-1"
[routerLink]="['/products',product.id]">View</a> </td>
        </tr>
    </tbody>
```

Categorypost.component

Prihvatanje podataka iz forme za creiranje nove kategorije, vaidacija podataka upotrebom ugradjenih angular biblioteka, testiranje validosti forme i slanje podataka na servis za kreiranje k

ategorije.

```
export class CategorypostComponent implements OnInit {
  public categoryForm!:FormGroup;

  constructor(
    private categoryService:CategoriesService,
    private formbuilder:FormBuilder) {
    this.createForm();
   }

  public createForm(){
    this.categoryForm = this.formbuilder.group({
      name:['',[Validators.required]],
      image:['',[Validators.required]],
    });

  }

   public submitForm(formdata:PostCategory){
    if(!this.categoryForm.valid){
      window.alert("Form is not valid.");
      return;
    }else{

      this.categoryService.createCatogory(formdata)
      .subscribe((par:PostCategory)=>{
        window.alert(`Your category with name ${par.name} is created.`);
        window.location.href="categories";
      })

    }
  }
  ngOnInit(): void {
  }
}
```

Categoryupdate.component

Obnova izabrone kategorije na osnovu id variable is url http string-a, inicijano popujivanje forme sa podacima kategorije za obnovu, validacija ispravljenih podataka, slanje pdataka servisu za update kategorije.

```
export class CategoryupdateComponent implements OnInit {
  public id!:string|null;
  public category!:Category;
  public editForm!:FormGroup;

  constructor(private route:ActivatedRoute,
    private catService:CategoriesService,
    private formbuilder:FormBuilder) {

    this.route.paramMap.subscribe( par => {
      this.id = par.get('id');
    });

    this.catService.getCategoryById(this.id)
    .subscribe((e:Category) => {
      this.category = e;
      this.editForm.patchValue({
        name: e.name,
     });
      });
       this.createForm();
  }
  public createForm(){
    this.editForm = this.formbuilder.group({
      name:['',[Validators.required]],
    });
  }
  public submitForm(formdata:UpdateCategory){
    if(!this.editForm.valid){
      window.alert("Form is not valid.");
      return;
    }else{
      this.catService.editCategory(this.id, formdata)
      .subscribe((par:UpdateCategory)=>{
        window.alert(`Your category was updated.`);
        window.location.href='categories';
      })

    }
  }

  ngOnInit(): void {
```

```
  }

}
```

Cover.component

Prikaz nekih od elemenata na cover stranici sajta.

```
export class CoverComponent implements OnInit {
  data:any;
  constructor(private productService:PostService) {
    this.productService.refreshProducts()
    .subscribe((e)=>{
      this.data = e;
    });
   }

  ngOnInit(): void {
  }

}
```

Login.component

Hard code setovanje login elemenata u formi logovanja za brzi proces autentifikacije.

Slanje podataka servisu za dalju obradu, i obavestenje korisniku ako je komunikacija sa api-jem uspesna. Postavljanje dobijenog token elementa sa servera u localStorage za potrebe aplikacije.

```
export class LoginComponent implements OnInit {
   logForm!: FormGroup;
   email!:string;
   password!:string;
   constructor(private authService : AuthService, private router : Router) { }

   ngOnInit() {
      this.logForm = new FormGroup({
         email: new FormControl(),
         password:new FormControl()
     });
     this.logForm.patchValue({
      email: "john@mail.com",
      password:"changeme"
   });
   }
```

```
  onClickSubmit(data:any) {
     this.email = data.email;
     this.password = data.password;

     this.authService.login(this.email,this.password)
        .subscribe( (data:TokenData) => {
         if(data){
            localStorage.setItem("token",data.access_token);
           window.location.href="/productslist";
         }
     });

  }
}
```

Logout.component

Pozivanje logout motode iz servisa i redirekcija korisnika nakon uspesne odjave.

```
export class LogoutComponent implements OnInit {

  constructor(private authService : AuthService, private router: Router) { }

  ngOnInit() {
     this.authService.logout();
     window.location.href="/products";
  }

}
```

Models

Razni modeli za prenos podataka u aplikaciji. Vecina njih se reflektuje na nacin slanja podataka enpoint-ovima za laksi prenos i manipulaciju.

Navigation.component

Navigacija sajta sa proverom akcije logovanja korisnika, odnosno cuvanja ispravnog tokena sa servera u aplikaciji. Uzavisnosti od te akcije imamo smenu pojavljivanja linkova u navigaciji.

```
export class NavigationComponent implements OnInit {

  isUserLoggedIn = false;

  constructor(private authService: AuthService) {}

  ngOnInit() {
     let storeData = localStorage.getItem("token");
     if( storeData != null)
        this.isUserLoggedIn = true;
     else
        this.isUserLoggedIn = false;
  }
}
```

Productinfo.component

Prikaz proizvoda na zasebnoj strani sa smenom linkova na stani za akcije obnove i brisanja proizvoda.

Poziv ka proizvod servisu za brisanje proizvoda.

```
export class ProductInfoComponent implements OnInit {
   public product!: InTwineProduct;
  public id!:string|null;
  public isUserLoggedIn:boolean = false;
  constructor(private postservice:PostService,
    private route:ActivatedRoute,) {

    this.route.paramMap.subscribe( par => {
      this.id = par.get('id');
    })
     this.postservice.getProductById(this.id)
      .subscribe((e:InTwineProduct) => {
        this.product = e;
        });
   }
```

```
  ngOnInit(): void {
    let storeData = localStorage.getItem("token");
     if( storeData != null)
        this.isUserLoggedIn = true;
     else
        this.isUserLoggedIn = false;
  }
  public deleteProduct(){
    this.postservice.destroyProduct(this.id)
    .subscribe((e)=>{
      window.alert("Your product has been removed.");
      window.location.href = '/';
    })
  }

}
```

Productlist.component

Prikaz svih proizvoda na list stranici uz mogucnost upotrebe jednostavnje paginacije.

```
export class ProductListComponent implements OnInit {
  data:any;
  page:number = 1;

  constructor(private productService: PostService) {  }
    ngOnInit(): void {
      this.getData();
    }
    public getData(){
      this.productService.refreshProducts()
      .subscribe((e)=>{
        this.data = e;
      });
    }


}
```

```
  <div class="py-5 bg-light">
        <div class="row">
            <div *ngFor="let pro of data| paginate: {
            itemsPerPage: 10,
             currentPage: page }" class="col-md-4">
```

```
                <div class="card mb-4 box-shadow">
                    <div *ngFor="let image of pro.images; index as i; first as
isFirst">
                        <img class="d-block w-100" *ngIf="isFirst" src="{{
image }}" />
                    </div>
                    <div class="card-body">
                        <h5 class="card-title">{{ pro.title }}</h5>
                        <h6 class="card-subtitle mb-2 text-muted">{{
pro.price|currency }}</h6>
                        <p class="card-text">{{ pro.description }}</p>
                        <h6 class="card-subtitle mb-2 text-muted">{{
pro.category.name }}</h6>
                        <br>
                        <a class="btn btn-primary"
[routerLink]="['/products',pro.id]">See Details</a>
                    </div>
                </div>

            </div>

        </div>
        <div class="row">
            <div class="col-6 offset-3">
                <pagination-controls (pageChange)="page =
$event"></pagination-controls>
            </div>
        </div>

    </div>
```

Productpost.component

Prikaz forme za kreiranje novih proizvoda uz odabir postojecih kategorija iz list na strani.

Validacija poslativ podataka i slanje servisu za dalju obradu.

```
export class ProductPostComponent implements OnInit {
  public productForm!:FormGroup;
  categories!:Observable<Category[]>;

  constructor(private postService:PostService,
    private categoryService:CategoriesService,
    private formbuilder:FormBuilder) {
    this.getCategories();
```

```
    this.createForm();
   }

   public getCategories(){
    this.categories = this.categoryService.getCategories();
   }

  public createForm(){
    this.productForm = this.formbuilder.group({
      title:['',[Validators.required]],
      price:['',[Validators.required]],
      description:['',[Validators.required]],
      categoryId:['',[Validators.required]],
      images:['',[Validators.required]],
    });

  }

   public submitForm(formdata:PostProduct){
    if(!this.productForm.valid){
      window.alert("Form is not valid.");
      return;
    }else{
      this.postService.createProduct(formdata)
      .subscribe((par:PostProduct)=>{
        window.alert(`Your product with title ${par.title} is created.`);
        this.productForm.reset();
      })

    }
  }
  ngOnInit(): void {
  }

}
```

Productupdate.component

Inicijano popunjavanje forme za obnovu podataka proizvoda, validacija poslatih podataka i obavestavanje korisnika o stanju akcije.

```
export class ProductUpdateComponent implements OnInit {
  public editForm!:FormGroup;
  public id!:string|null;
  public product!: InTwineProduct;

  constructor(private route:ActivatedRoute,
    private postService:PostService,
```

```
    private formbuilder:FormBuilder ) {

      this.route.paramMap.subscribe( par => {
        this.id = par.get('id');
      });
      this.postService.getProductById(this.id)
      .subscribe((e:InTwineProduct) => {
        this.product = e;
        this.editForm.patchValue({
          title: e.title,
          price: e.price
       });
        });
      this.createForm();
   }

  public createForm(){
    this.editForm = this.formbuilder.group({
      title:['',[Validators.required]],
      price:['',[Validators.required]],
    });

  }

   public submitForm(formdata:UpdateProduct){
    if(!this.editForm.valid){
      window.alert("Form is not valid.");
      return;
    }else{
      this.postService.editProduct(this.id, formdata)
      .subscribe((par:UpdateProduct)=>{
        window.alert(`Your product with title ${par.title} was updated.`);
        this.editForm.reset();
      })

    }
  }

  ngOnInit(): void {
  }

}
```

Services

Auth.service

Odradjivanje login i logout fungcionalnosti i komunikacija sa fake api serverom.

```
export class AuthService {
  public url ='https://api.escuelajs.co/api/v1/auth/login';

   constructor(private http:HttpClient) { }

   public login(email: string, password: string){

      const body = { email,password }
      console.log(body);

      return this.http.post<any>(this.url, body);
   }

   public logout(): void {
       localStorage.removeItem('token');
    }

}
```

Categories.service

Metode koje manipulisu endpoint-ovima promenom isith i slanje lil dohvatenje odredjenih podataka vezanih za kategorije proizvoda.

```
export class CategoriesService {

  readonly url = 'https://api.escuelajs.co/api/v1/categories';
  constructor(private http:HttpClient) {}

   public getCategories(){
   return this.http.get<Category[]>(this.url);
   }
   public getCategoryById(id:string|null){
    return this.http.get<Category>(this.url+'/'+id);
   }
   public getProductsbyCategory(id:string|null){
    return this.http.get<InTwineProduct[]>(this.url+'/'+id+'/products');

   }
   public editCategory(id:string|null, param:UpdateCategory){
    const body = param;
    return this.http.put<any>(this.url +'/'+ id, body)
  }
```

```
  public createCatogory(formdata:PostCategory){
    const body = formdata;
    console.log(body);
    return this.http.post<PostCategory>(this.url, body);
  }
}
```

Post.service

Metode koje manipulisu endpoint-ovima promenom isith i slanje lil dohvatenje odredjenih podataka vezanih za prizvode.

```
export class PostService {

  readonly url = 'https://api.escuelajs.co/api/v1/products';
  constructor(private http:HttpClient) {}

   public refreshProducts(){
   return this.http.get(this.url);
   }

   public getProductById(id:string|null){
   return this.http.get<InTwineProduct>(this.url +'/'+ id);
   }

   public createProduct(formdata:PostProduct):Observable<PostProduct>{
        const body ={
          title:formdata.title,
          price:formdata.price,
          description:formdata.description,
          categoryId: Number(formdata.categoryId),
          images:[formdata.images]
        }
     return this.http.post<PostProduct>(this.url, body);
    }
    public destroyProduct(id:string|null){
      return this.http.delete<any>(this.url+"/"+id);

    }
    public editProduct(id:string|null, param:UpdateProduct){
      const body = param;
      return this.http.put<any>(this.url +'/'+ id,body)
    }



}
```

App-routing.module

Integracija ruting modula koji komunicira sa svim komponentima aplikacije,

Definisanje rute aplikacije, pristupne korisnicima u zavisnosti od Authguard dozvole, koja vrsi autorizaciju za odredjenje rute.

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { CategoryComponent } from './category/category.component';
import { CategoryinfoComponent } from './categoryinfo/categoryinfo.component';
import { CategorypostComponent } from './categorypost/categorypost.component';
import { CategoryupdateComponent } from
'./categoryupdate/categoryupdate.component';
import { LoginComponent } from './login/login.component';
import { LogoutComponent } from './logout/logout.component';
import { ProductInfoComponent } from './product-info/product-info.component';
import { ProductListComponent } from './product-list/product-list.component';
import { ProductPostComponent } from './product-post/product-post.component';
import { ProductUpdateComponent } from './product-update/product-
update.component';

import { AuthGuard } from './auth.guard';
import { CoverComponent } from './cover/cover.component';

const routes: Routes = [
  { path: '', component: CoverComponent },
  { path: 'productslist', component: ProductListComponent },
  { path: 'addproduct', component: ProductPostComponent ,canActivate:
[AuthGuard]},
  { path: 'products/:id', component: ProductInfoComponent },
  { path: 'editproduct/:id', component: ProductUpdateComponent ,canActivate:
[AuthGuard]},

  { path: 'categories', component: CategoryComponent },
  { path: 'addcategories', component: CategorypostComponent ,canActivate:
[AuthGuard]},
  { path: 'categoryinfo/:id', component: CategoryinfoComponent },
  { path: 'editcategories/:id', component: CategoryupdateComponent
,canActivate: [AuthGuard]},

  { path: 'login', component: LoginComponent },
  { path: 'logout', component: LogoutComponent },

];

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

```
export class AppRoutingModule { }
```

auth.guard

Guard element za prepoznavanje zadate url adrese i redirekciju korisnika u zavisnosti od stanja autorizacije (postajanja token elementa u aplikaciji).

```
import { Injectable } from '@angular/core';
import { CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot, Router,
UrlTree } from '@angular/router';
import { Observable } from 'rxjs';

import { AuthService } from './services/auth.service';

@Injectable({
   providedIn: 'root'
})
export class AuthGuard implements CanActivate {

   constructor(private authService: AuthService, private router: Router) {}

   canActivate(
   next: ActivatedRouteSnapshot,
   state: RouterStateSnapshot): boolean | UrlTree {
      let url: string = state.url;

          return this.checkLogin(url);
      }

      checkLogin(url: string): true | UrlTree {
         let val: string|null = localStorage.getItem('token');

         if(val == null){
          return this.router.parseUrl('/login');
         }
         return true;

      }
}
```