









**HYPER COMPANY
FRONT-END
ANGULAR BOOTCAMP**



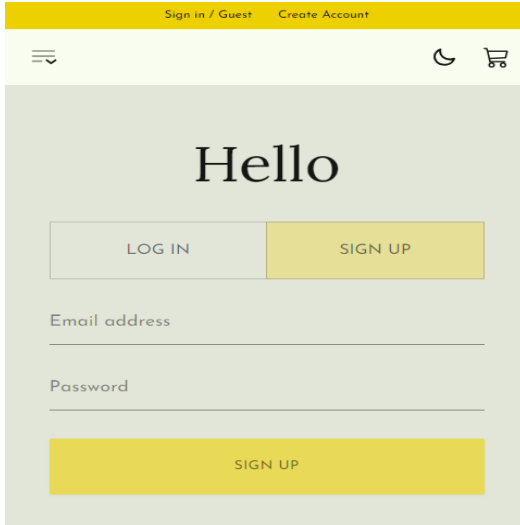
Serhat KARADAĞ

**ANGULAR
E-TİCARET
BİTİRME PROJESİ**

KULLANILAN TEKNOLOJİLER

- Frontend :  Angular
- Styling :  Tailwind CSS  daisyUI
- Form Management :  Angular Reactive Forms
- State Management :  NgRx  Angular Service
- Backend :  Node.js
- Database :  MongoDB

Light - Dark Mode



DaisyUI sınıfından renk temaları seçilir.

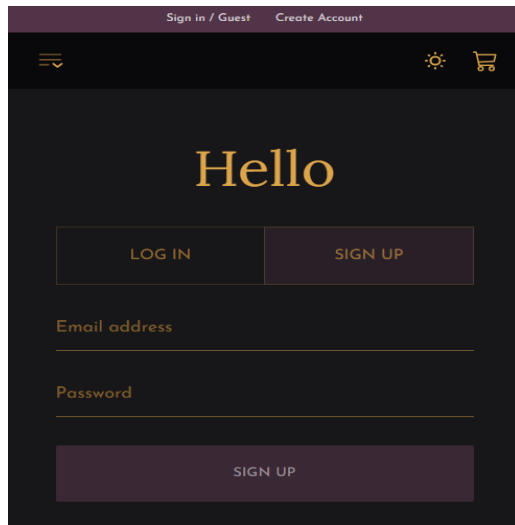


```
const themes = {  
  lemonade: 'lemonade',  
  luxury: 'luxury',  
};
```

Angular'ın 'signal' özelliği kullanılarak state kontrol edilir ve local storage'a kaydedilir.



```
theme = signal<string>(this.getThemeFromLocalStorage());  
  
getThemeFromLocalStorage() {  
  return localStorage.getItem('theme') || themes.lemonade;  
}  
  
constructor() {  
  effect(() => {  
    document.documentElement.setAttribute('data-theme', this.theme());  
    localStorage.setItem('theme', this.theme());  
  });  
}
```



Kullanıcı icon'a tıkladığında tema değişir.



```
<input (change)="handleTheme()" />
```

```
handleTheme() {  
  const { lemonade, luxury } = themes;  
  const newTheme = this.theme() === lemonade ? luxury : lemonade;  
  this.theme.set(newTheme);  
}
```

NgRx ile Global State Management

NgRx ile state'i globalde oluşturup istediğimiz componentin içerisinde çağırabiliriz.

Aksiyon farklılıklarına göre kendi oluşturduğumuz state'lerin değerlerini değiştiririz.

State değişikliklerine göre sayfaya farklı componentleri getiririz.

```
const initialState = {  
  isLoading: false,  
  data: null,  
};
```

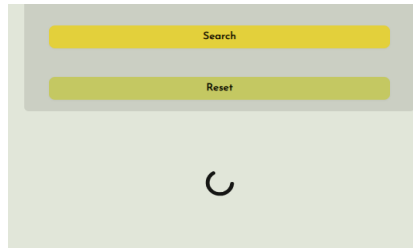
```
reducer: createReducer(  
  initialState,  
  on(productsActions.getProducts, (state) => ({  
    ...state,  
    isLoading: true,  
  })),  
  on(productsActions.getProductsSuccess, (state, action) => ({  
    ...state,  
    isLoading: false,  
    data: action.products,  
  })),  
),
```

```
@if(isLoading) {  
  <eCommerce-loading/>  
} @else if (data) {  
  <eCommerce-productContainer />  
}
```

Kullanıcı /products sayfasına gittiğinde

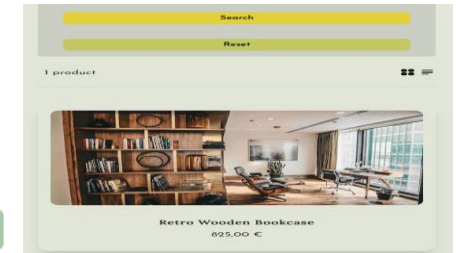
getProducts
aksiyonu tetiklenir.

isLoading (pin): false => true

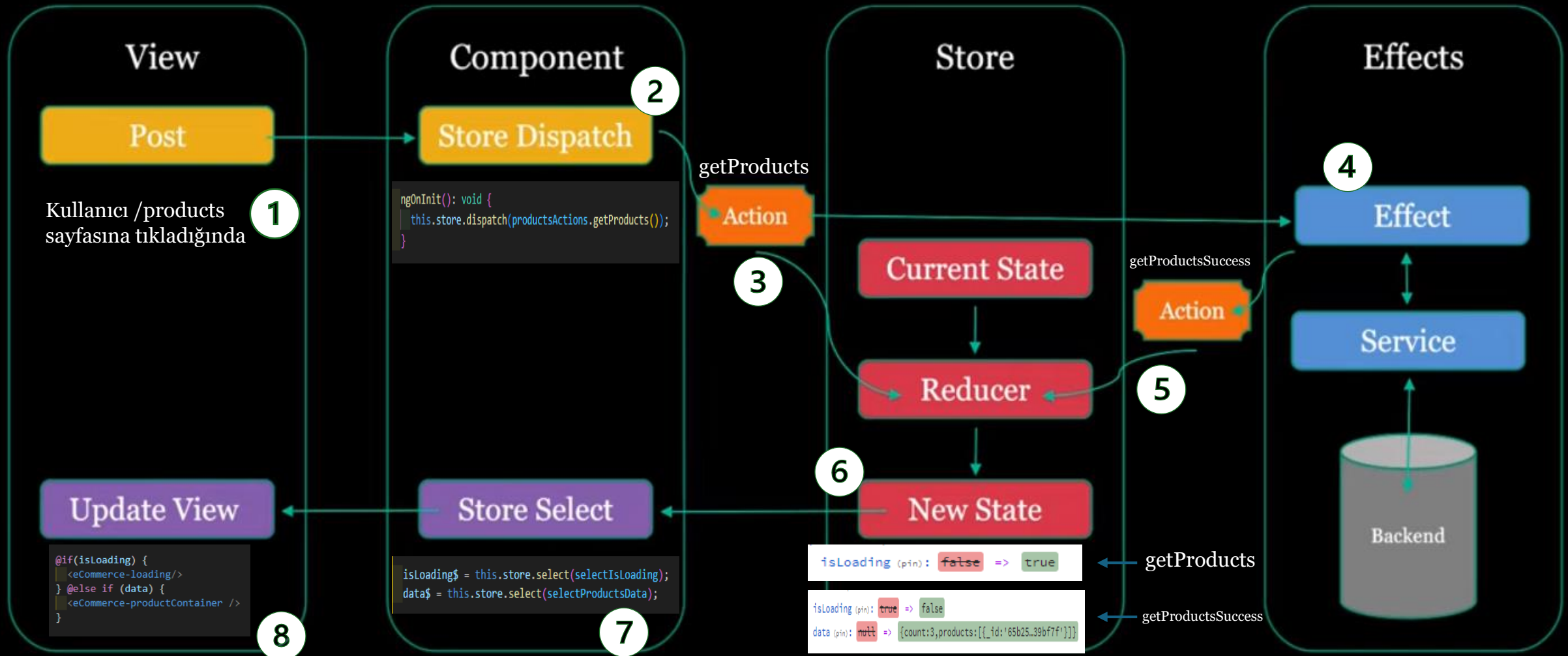


getProductsSuccess
aksiyonu tetiklenir.

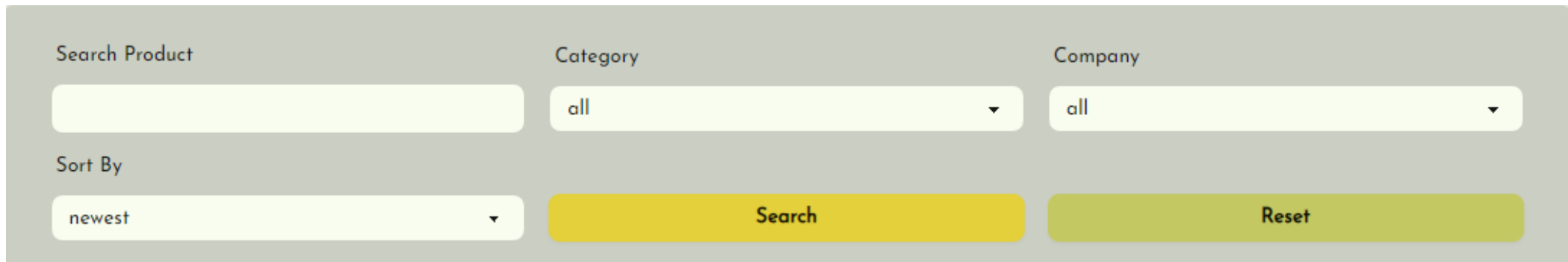
isLoading (pin): true => false
data (pin): null => {count:3,products:[{_id:'65b25...39bf7f'}]}



NgRx ile Global State Management



Filter Component



```
<a (click)="onClickButton()">Search</a>
```

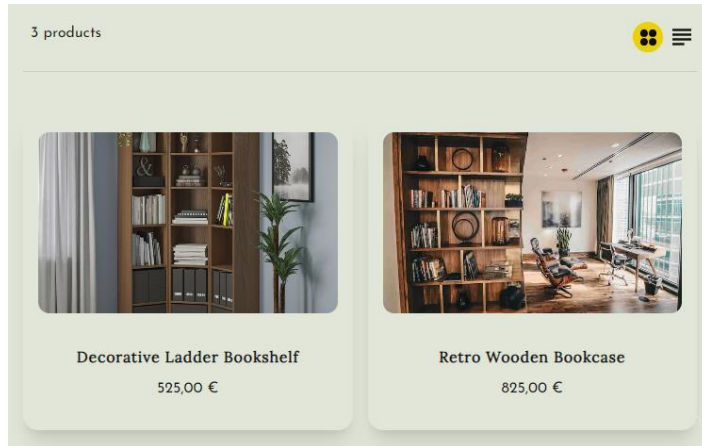
```
onClickButton() {  
  this.url = `/products/?search=${this.search}&category=${this.category}&company=${this.company}&sort=${this.sort}`;  
  this.store.dispatch(productsActions.getProducts({ url: this.url }));  
}
```

Kullanıcı 'search' butonuna bastığında, query parametreleriyle birlikte backend'e istek gidecek.

```
const { search, category, company, sort } = req.query;  
const queryObject = {};  
  
if (category && category !== "all") {  
  queryObject.category = category;  
}  
  
const products = await Product.find(queryObject);
```

Backend tarafında ise request'den gelen query parametrelere göre database'den bilgi çekilip geri döndürecek.

Grid / List Component



‘Signal’ özelliği sayesinde state kontrol edilir ve kullanıcı tıkladığı ikona göre ekranda grid veya list layout düzenini görür.

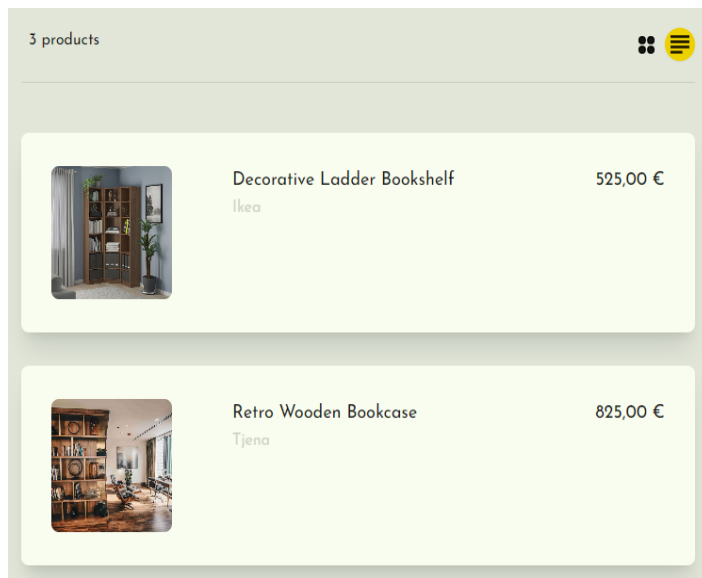


```
layout = signal('grid');

setList() {
  this.layout.set('list');
}

setGrid() {
  this.layout.set('grid');
}
```

```
@if(layout() === "grid") {
  <eCommerce-productsGrid />
} @else {
  <eCommerce-productsList />
}
```



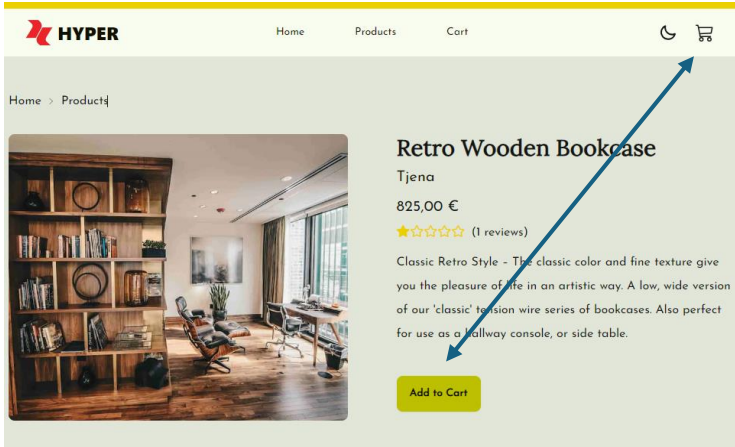
‘NgClass’ özelliği sayesinde aktif olan layout ikonunun arka plan rengi değişir.



```
setActiveStyles(pattern: string) {
  return `btn btn-circle btn-sm ${
    pattern === this.layout()
      ? 'btn-accent text-accent-content'
      : 'btn-ghost text-based-content'
  }`;
}
```

```
<button [ngClass]="setActiveStyles('grid')" (click)="setGrid()">
<button [ngClass]="setActiveStyles('list')" (click)="setList()">
```

Service ve BehaviorSubject ile Global State Management



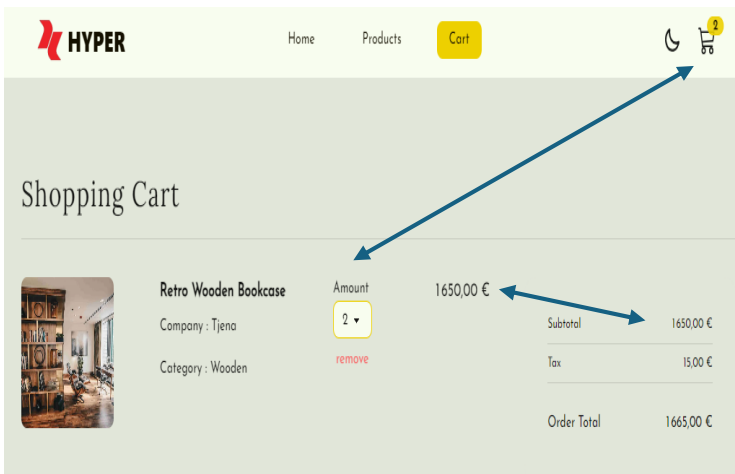
Cart bilgilerini localStorage'a kaydedip BehaviorSubject'e başlangıç değeri olarak atıyorum.

'Next' fonksiyonu sayesinde cart bilgileri her değiştiğinde bu observable'ı dinleyen herkes güncel datayı çeksin isteriz.

```
private cart: Cart = this.getCartFromLocalStorage();
private cartSubject: BehaviorSubject<Cart> = new BehaviorSubject(this.cart);

private setCartToLocalStorage(): void {
  localStorage.setItem('cart', JSON.stringify(this.cart));
  this.cartSubject.next(this.cart);
}

addToCart(bookcase: Bookcase): void {
  this.cart.items.push(new CartItem(bookcase));
  this.setCartToLocalStorage();
}
```



Component içerisinde CartService çağırılır ve click eventiyle birlikte state güncellenir.

```
<button (click)="addToCart()">Add to Cart</button>
```

```
addToCart() {
  this.cartService.addToCart(this.bookcase);
  this.router.navigateByUrl('/cart');
}
```


Reactive Forms ile Form Management

The image shows a web form with a light green background. At the top, it says 'Hello' in a large, dark font. Below that, there's a red error message 'Provide valid email'. There are two buttons: 'LOG IN' (light green) and 'SIGN UP' (yellow). Below the buttons, there's a text input field containing 'serhatKaradağ', followed by a password input field with six dots. At the bottom, there's a large yellow button labeled 'SIGN UP'.

'formControlName' özelliği sayesinde inputlar kontrol edilir. Kullanıcının gerekli alanları doldurması zorunlu kılınır.

```
form = this.fb.nonNullable.group({  
  email: ['', Validators.required],  
  password: ['', Validators.required],  
});
```

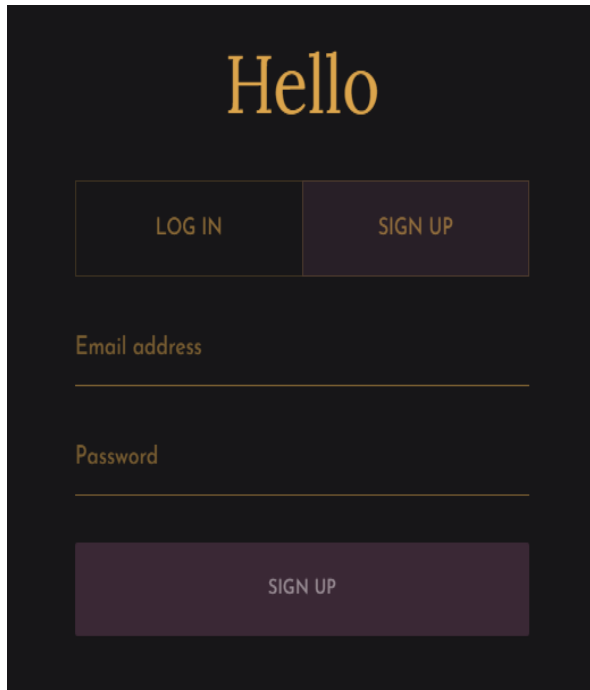
Form, invalid olduğu sürece button elementi disabled yapılır.

```
<button [disabled]="form.invalid || (isSubmitting$ | async)">
```

Kullanıcı submit yaptığında 'getRawValue' fonksiyonuyla inputtan bilgiler alınır ve API isteği yapılır.

```
onSubmit() {  
  const request = this.form.getRawValue();  
  this.store.dispatch(authActions.register({ request }));  
}
```

Authentication



Hello

LOG IN SIGN UP

Email address

Password

SIGN UP

Backend tarafında JWT yaratılır ve response olarak kullanıcıya gönderilir. Gelecek isteklerde request'in header'ı kontrol edilir.

```
const authHeader = req.headers.authorization;

if (authHeader && authHeader.startsWith("Bearer")) {
  token = authHeader.split(" ")[1];
}

if (!token) {
  throw new CustomError.UnauthenticatedError("Authentication invalid");
}
```

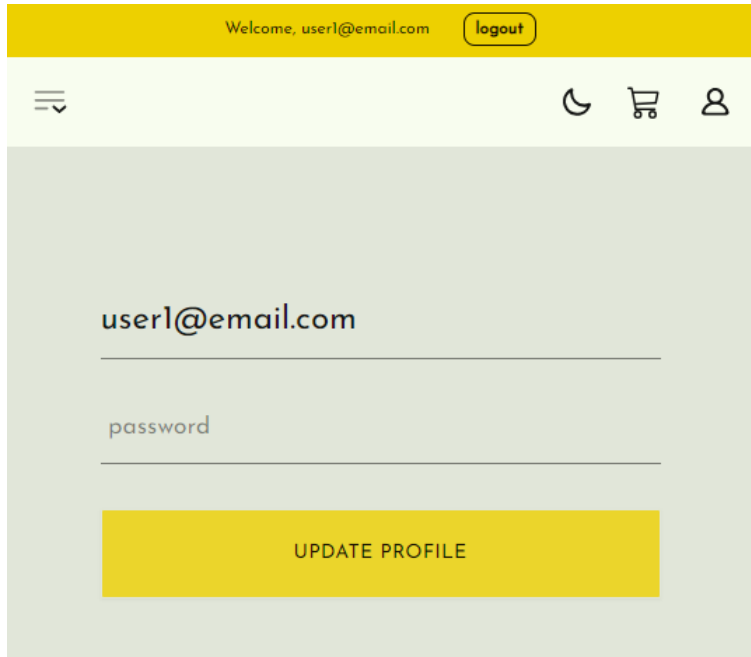
Frontend tarafında kullanıcı login veya register olduğunda NgRx'in 'effect' fonksiyonu içinde token localStorage'a kaydedilir.

```
map((currentUser) => {
  localStorageService.set('accessToken', currentUser.token);
  return authActions.loginSuccess({ currentUser });
}),
```

Interceptor aracılığıyla request'in header'ına token eklenir.

```
const localStorageService = inject(LocalStorageService);
const token = localStorageService.get('accessToken');
request = request.clone({
  setHeaders: {
    Authorization: token ? `Bearer ${token}` : '',
  },
});
return next(request);
```

User Profile Component



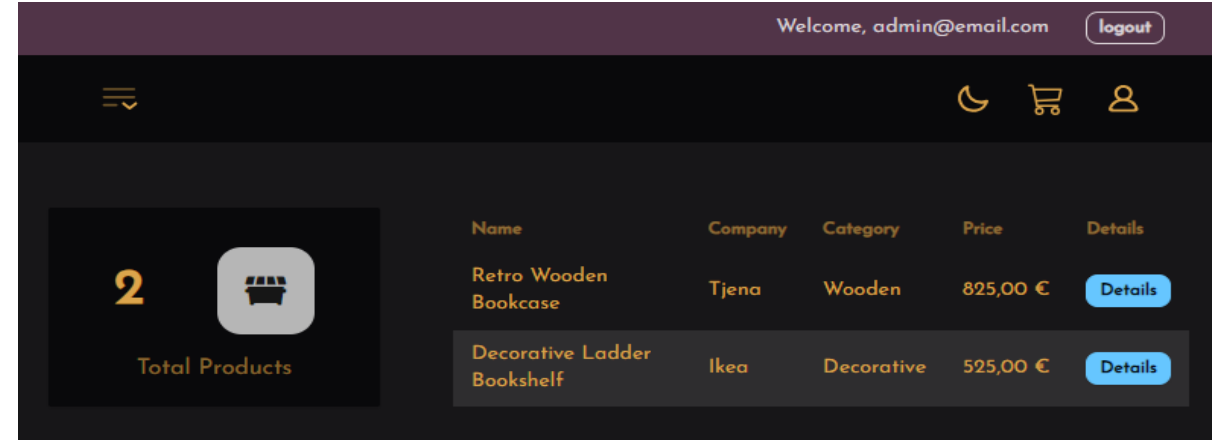
The image shows a web application interface for a user profile. At the top, a yellow header bar contains the text "Welcome, user1@email.com" and a "logout" button. Below the header, a navigation bar includes a hamburger menu icon, a refresh icon, a shopping cart icon, and a user profile icon. The main content area has a light gray background and contains a form with two input fields: "user1@email.com" and "password". Below these fields is a yellow button labeled "UPDATE PROFILE".

Kullanıcı /profile sayfasına gittiğinde bilgiler otomatik olarak dolu bir şekilde gelir. Çünkü 'filter' RxJS fonksiyonu sayesinde current user 'undefined' veya 'null' olduğu sürece filter'dan geçemez ve initializeForm fonksiyonu başlamaz.

```
ngOnInit(): void {  
  this.store  
    .pipe(select(selectCurrentUser), filter(Boolean))  
    .subscribe((currentUser) => {  
      this.currentUser = currentUser;  
      this.initializeForm();  
    });  
}  
  
initializeForm(): void {  
  this.form.patchValue({  
    email: this.currentUser?.email,  
    password: '',  
  });  
}
```

Admin Dashboard ve CRUD işlemleri

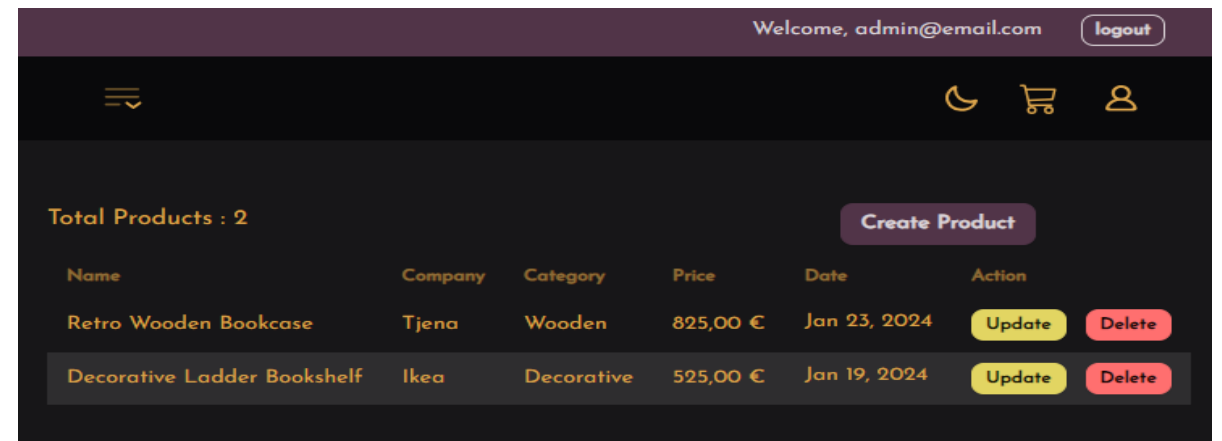
Admin Dashboard sayfasında ürün bilgileri NgRx store'dan güncel bir şekilde alınır ve tablo halinde gösterilir.



The screenshot shows the Admin Dashboard. At the top, there's a header with 'Welcome, admin@email.com' and a 'logout' button. Below the header, there's a navigation bar with a menu icon, a moon icon, a shopping cart icon, and a user icon. The main content area features a 'Total Products' card on the left showing '2' and a shopping cart icon. To the right, there's a table with product information.

Name	Company	Category	Price	Details
Retro Wooden Bookcase	Tjena	Wooden	825,00 €	Details
Decorative Ladder Bookshelf	Ikea	Decorative	525,00 €	Details

Admin /products sayfasında CRUD işlemleri yapılır.

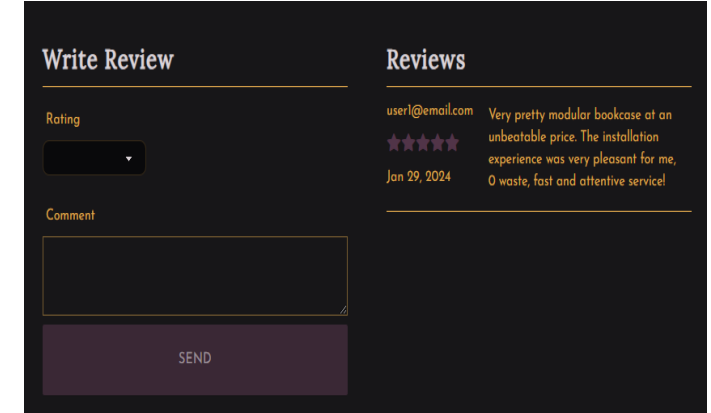
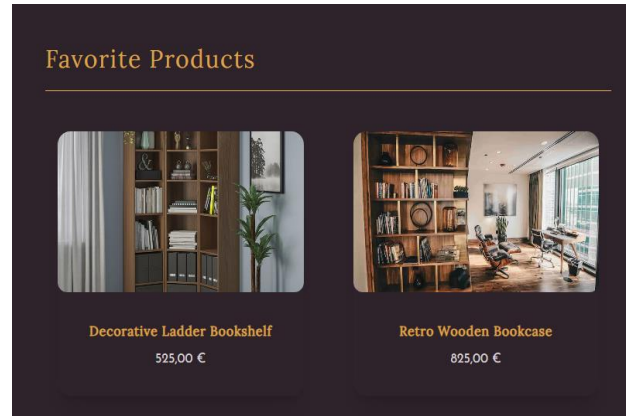


The screenshot shows the Admin /products page. At the top, there's a header with 'Welcome, admin@email.com' and a 'logout' button. Below the header, there's a navigation bar with a menu icon, a moon icon, a shopping cart icon, and a user icon. The main content area features a 'Total Products : 2' label and a 'Create Product' button. Below this, there's a table with product information and actions.

Name	Company	Category	Price	Date	Action
Retro Wooden Bookcase	Tjena	Wooden	825,00 €	Jan 23, 2024	Update Delete
Decorative Ladder Bookshelf	Ikea	Decorative	525,00 €	Jan 19, 2024	Update Delete

Reviews Component

Anasayfadaki Favorite Products componenti en yüksek rating'e göre sıralanıyor.



Kullanıcı bir ürüne yorum yaptığı zaman MongoDB'deki 'Aggregation Pipeline' özelliği sayesinde averageRating tekrar güncellenir ve anasayfadaki Favorite Products componenti güncel olarak en yüksek rating'e göre tekrar sıralanır.

```
const result = await this.aggregate([
  { $match: { product: productId } },
  {
    $group: {
      _id: null,
      averageRating: { $avg: "$rating" },
      numOfReviews: { $sum: 1 },
    },
  },
]);

await this.model("Product").findOneAndUpdate(
  { _id: productId },
  {
    averageRating: result[0].averageRating || 0,
    numOfReviews: result[0].numOfReviews || 0,
  }
);
```

HYPER COMPANY
FRONT-END
ANGULAR BOOTCAMP



Serhat KARADAĞ

BENİ
DİNLEDİĞİNİZ İÇİN
TEŞEKKÜRLER