

Webtechnológiák 2.

Féléves beadandó

Parfüméria webalkalmazás

Készítette:

Baffi Réka

F4IIYA

Gazdaságinformatikus BSc

1. Feladateleírás

A beadandóm egy parfümök nyilvántartó webalkalmazás.

Első lépésként a felhasználó be tud jelentkezni, hogy ezután használhassa az oldalt.

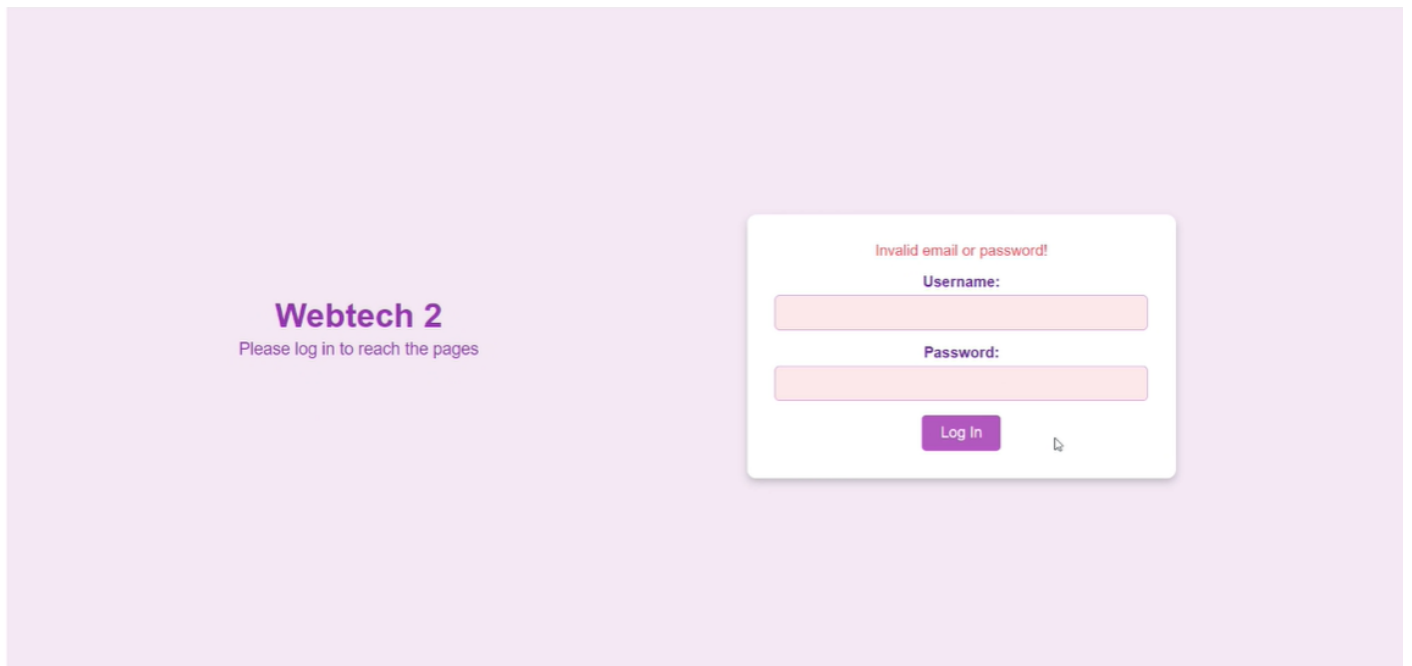
Belépés után a felhasználónak lehetősége van parfümök listázására, módosítására, törlésére illetve létrehozására.

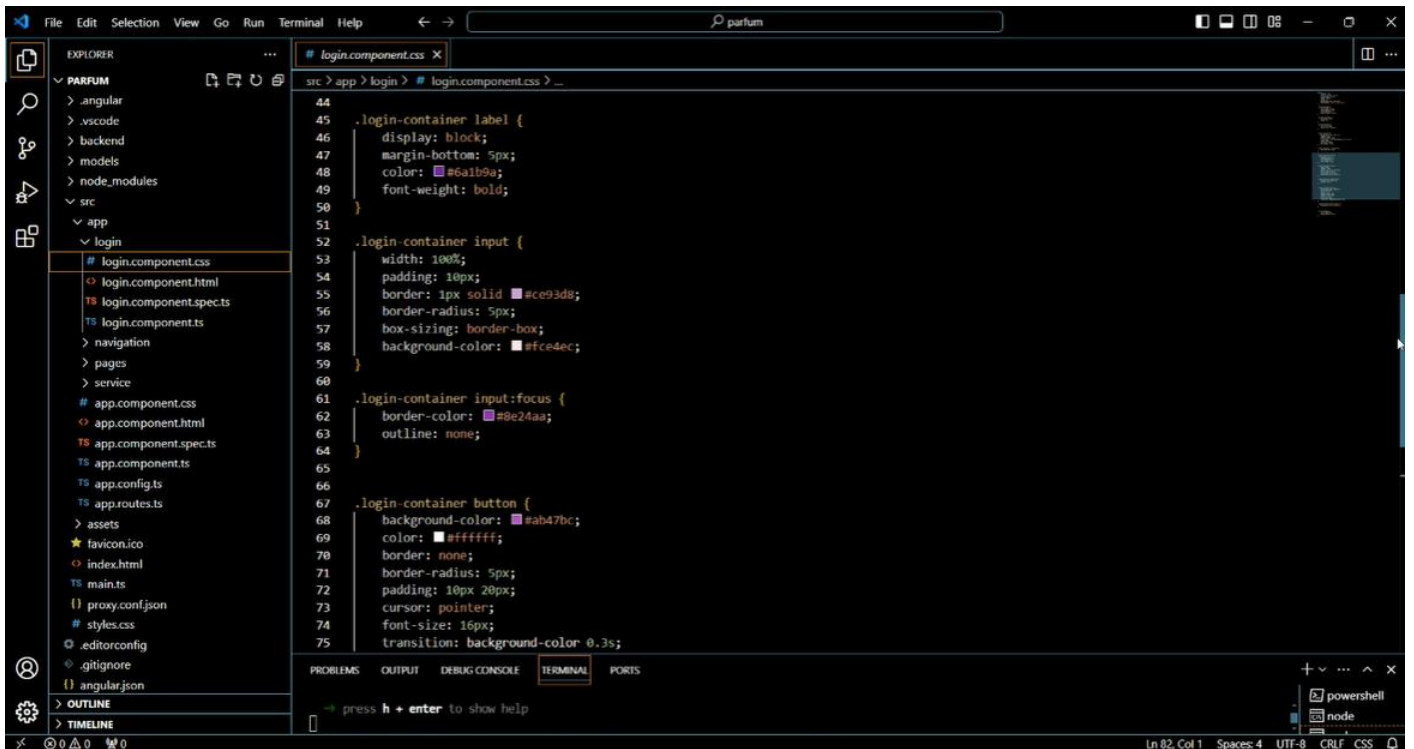
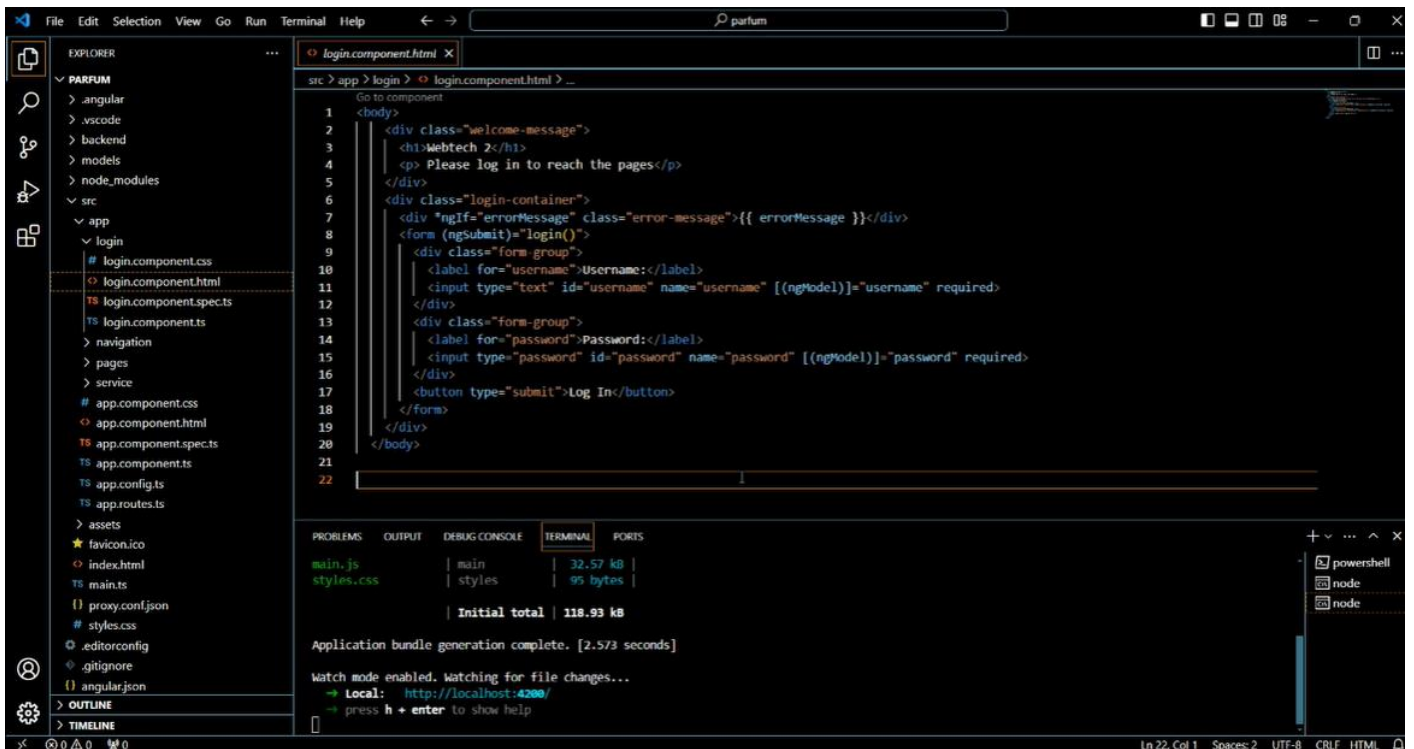
Végezetül, a felhasználó ki tud jelentkezni a rendszerből.

Az elkészített jegyzőkönyvemben részletesen bemutatom az elkészült alkalmazást, illetve annak forráskódjait.

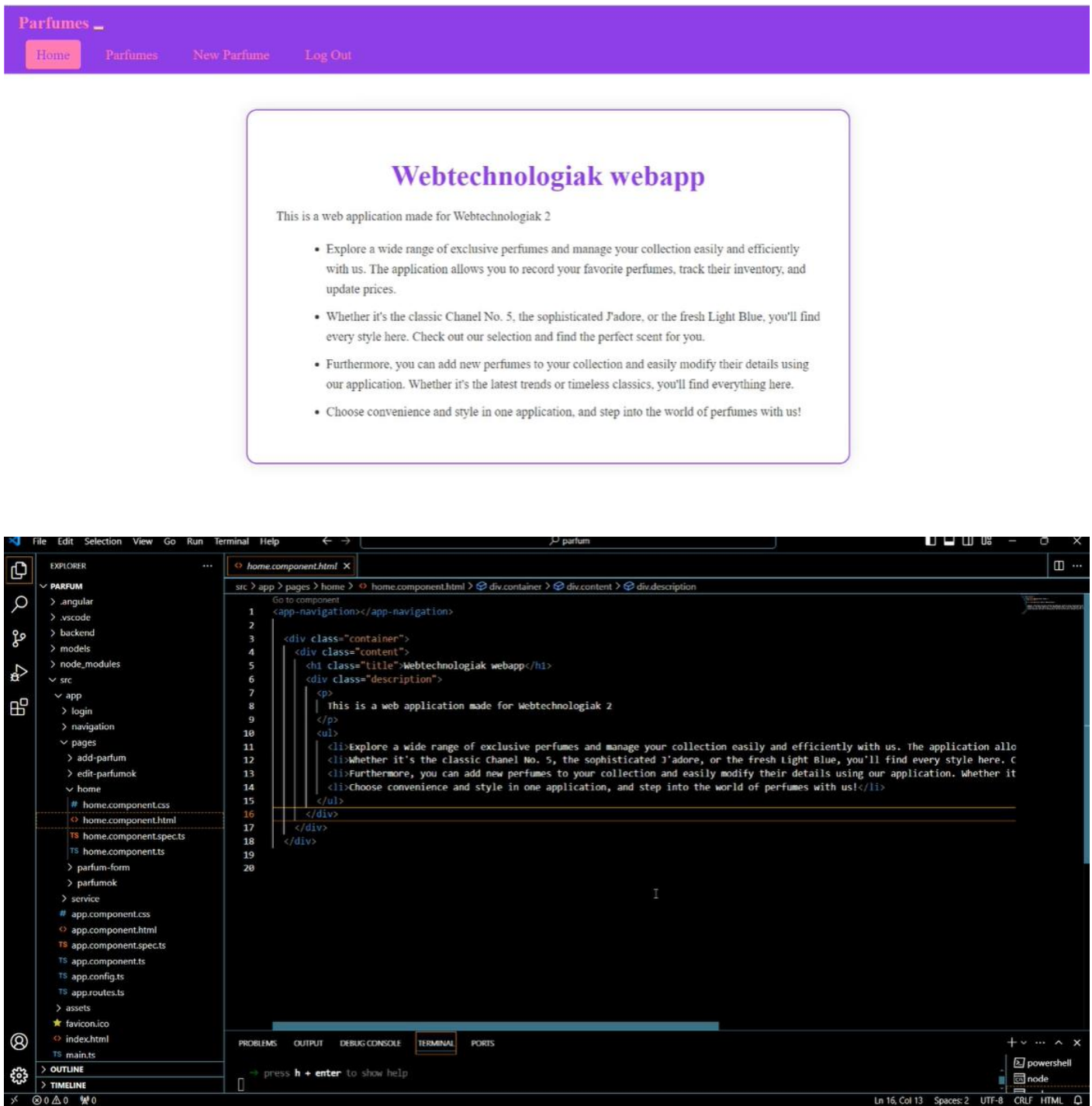
2. Bejelentkezés

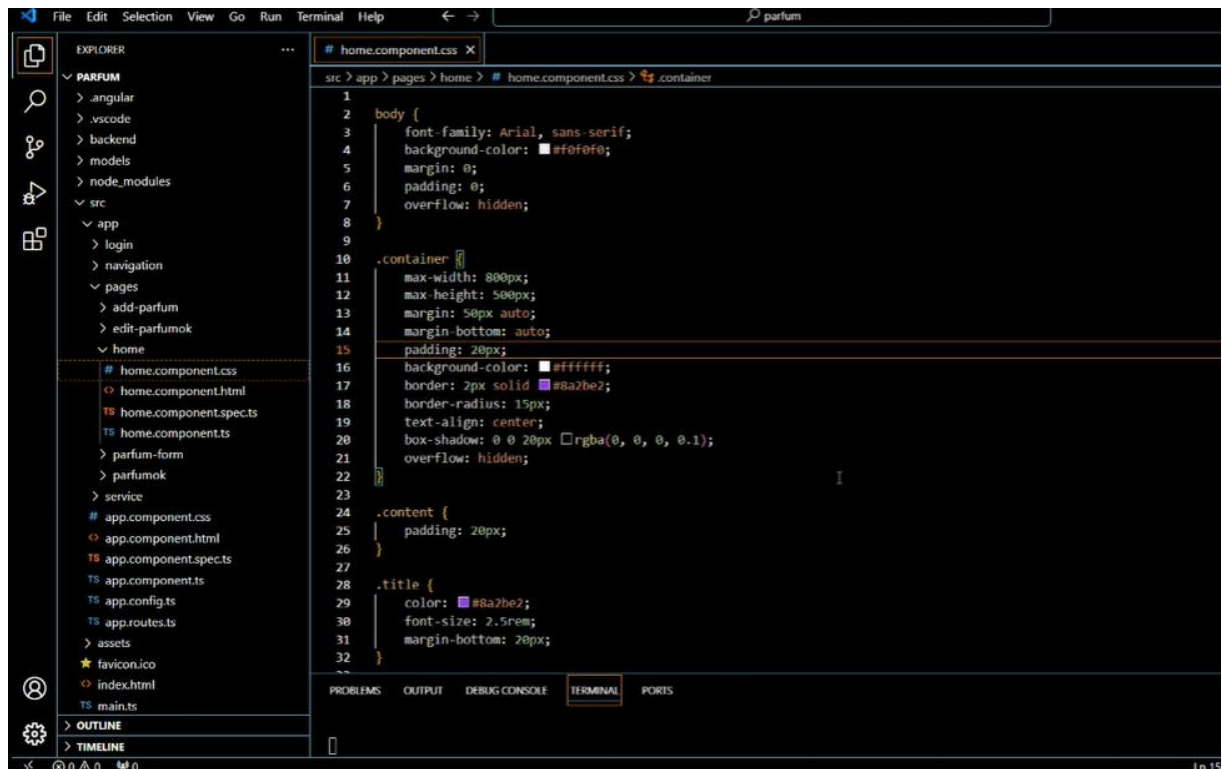
A komponens ellenőrzi a felhasználónév és jelszó helyességét





3. Főoldal





4. Parfümök listázása

Parfumes					
Home Parfumes New Perfume Log Out					
ID	Name	Brand	Price	Quantity	
21	Chanel No. 5	Chanel	100	50	Modify Delete
22	Padore	Dior	120	30	Modify Delete
23	Light Blue	Dolce & Gabbana	80	45	Modify Delete
24	La Vie Est Belle	Lancôme	95	40	Modify Delete
25	Flowerbomb	Viktor & Rolf	115	20	Modify Delete
26	Black Opium	Yves Saint Laurent	90	35	Modify Delete
27	Si	Giorgio Armani	85	50	Modify Delete
28	Angel	Thierry Mugler	100	25	Modify Delete
29	Guilty	Gucci	105	30	Modify Delete

5. Módosítás

ModifyDelete

Parfumes

[Home](#)[Parfumes](#)[New Perfume](#)[Log Out](#)

Name

Chanel No

Brand

Chanel

Price

100

Quantity

50

Save

Ha az 50-et átírjuk 25-re, akkor az adatbázisban megváltozik.

Parfumes

[Home](#)[Parfumes](#)[New Perfume](#)[Log Out](#)

Name

Chanel No

Brand

Chanel

Price

100

Quantity

25

Save

A megváltozott adatok:

ID	Name	Brand	Price	Quantity	
21	Chanel No	Chanel	100	25	ModifyDelete

6. Új rekord hozzáadása

Parfumes Home Parfumes New Parfume Log Out

Name

Brand

Price

Quantity

Save

Az új parfüm adatai megjelennek a táblázatban:

39	The One	Dolce & Gabbana	110	35	Modify Delete
----	---------	-----------------	-----	----	---

Forráskódok:

```
1 .table {
2   width: 100%;
3   margin-top: 20px;
4   border-collapse: separate;
5   border-spacing: 0;
6   background-color: #f0f0f0;
7   border: 2px solid #8a2be2;
8 }
9
10
11 .table thead {
12   background-color: #8a2be2;
13   color: #ffffff;
14 }
15
16 .table thead th {
17   padding: 15px;
18   text-align: left;
19   font-size: 1.2rem;
20   border-left: 2px solid #8a2be2;
21   border-top: 2px solid #8a2be2;
22   border-bottom: 2px solid #8a2be2;
23 }
24
25 .table thead th:first-child {
26   border-left: none;
27 }
28
29 .table tbody tr {
30   border-bottom: 2px solid #8a2be2;
31 }
32
```

This screenshot shows the Visual Studio Code editor with the Explorer sidebar on the left displaying the project structure. The main editor window shows the file `parfum-form.component.html` with the following HTML code:

```
1 <div class="container">
2   <div class="row">
3     <div class="col md 6">
4       <form #editForm="ngForm" [formgroup]="parfumForm">
5
6         <div class="mb-3">
7           <label for="name" class="form-label">Name</label>
8           <input required pattern="[a-zA-Z0-9\s]+" name="name" type="text" class="form-control" id="name"
9             [formcontrolname]="name">
10           @if(parfumForm.controls['name'].invalid) {
11             <div class="alert alert-danger mt-3">
12               Name is required
13             </div>
14           }
15         </div>
16
17         <div class="mb-3">
18           <label for="brand" class="form-label">Brand</label>
19           <input required pattern="[a-zA-Z0-9\s]+" name="brand" type="text" class="form-control"
20             id="brand" formcontrolname="brand">
21           @if(parfumForm.controls['brand'].invalid) {
22             <div class="alert alert-danger mt-3">
23               Brand is required
24             </div>
25           }
26         </div>
27
28         <div class="mb-3">
29           <label for="price" class="form-label">Price</label>
30           <input required pattern="[0-9]+$" name="price" type="text" class="form-control" id="price"
31             formcontrolname="price">
32           @if(parfumForm.controls['price'].invalid) {
33             <div class="alert alert-danger mt-3">
34               Price is required
35             </div>
36           }
37         </div>
38       </form>
39     </div>
40   </div>
41 </div>
```

This screenshot shows the Visual Studio Code editor with the Explorer sidebar on the left. The main editor window shows the file `parfum-form.component.ts` with the following TypeScript code:


```
1 import { Component } from '@angular/core';
2 import { Router } from '@angular/router';
3 import { ActivatedRoute } from '@angular/router';
4 import { FormBuilder } from '@angular/forms';
5 import { ParfumsService } from '../services/parfums.service';
6
7 @Component({
8   selector: 'app-parfum-form',
9   standalone: true,
10   imports: [
11     NavigationComponent,
12     ReactiveFormsModule,
13     CommonModule,
14     FormsModule
15   ],
16   templateUrl: './parfum-form.component.html',
17   styleUrls: ['./parfum-form.component.css']
18 })
19 export class ParfumFormComponent {
20
21   formBuilder = inject(FormBuilder);
22
23   parfumsService = inject(ParfumsService);
24
25   router = inject(Router);
26
27   activatedRoute = inject(ActivatedRoute);
28
29   parfumForm = this.formBuilder.group({
30     id: 0,
31     name: '',
32     brand: '',
33     price: 0,
34     quantity: 0
35   });
36
37   isNewParfum = true;
38
39   ngOnInit(): void {
40     this.id = this.activatedRoute.snapshot.paramMap.get('id');
41     if (this.id) {
42       this.isNewParfum = false;
43     }
44   }
45
46   onSubmit(): void {
47     if (this.parfumForm.invalid) {
48       return;
49     }
50     const { id, name, brand, price, quantity } = this.parfumForm.value;
51     if (this.isNewParfum) {
52       this.parfumsService.addParfum({ name, brand, price, quantity });
53     } else {
54       this.parfumsService.updateParfum(id, { name, brand, price, quantity });
55     }
56     this.router.navigate(['/parfums']);
57   }
58
59   onDelete(id: number): void {
60     this.parfumsService.deleteParfum(id);
61   }
62 }
```

Listázás html kódja:

This screenshot shows the Visual Studio Code editor with the Explorer sidebar on the left. The main editor window shows the file `parfumok.component.html` with the following HTML code:

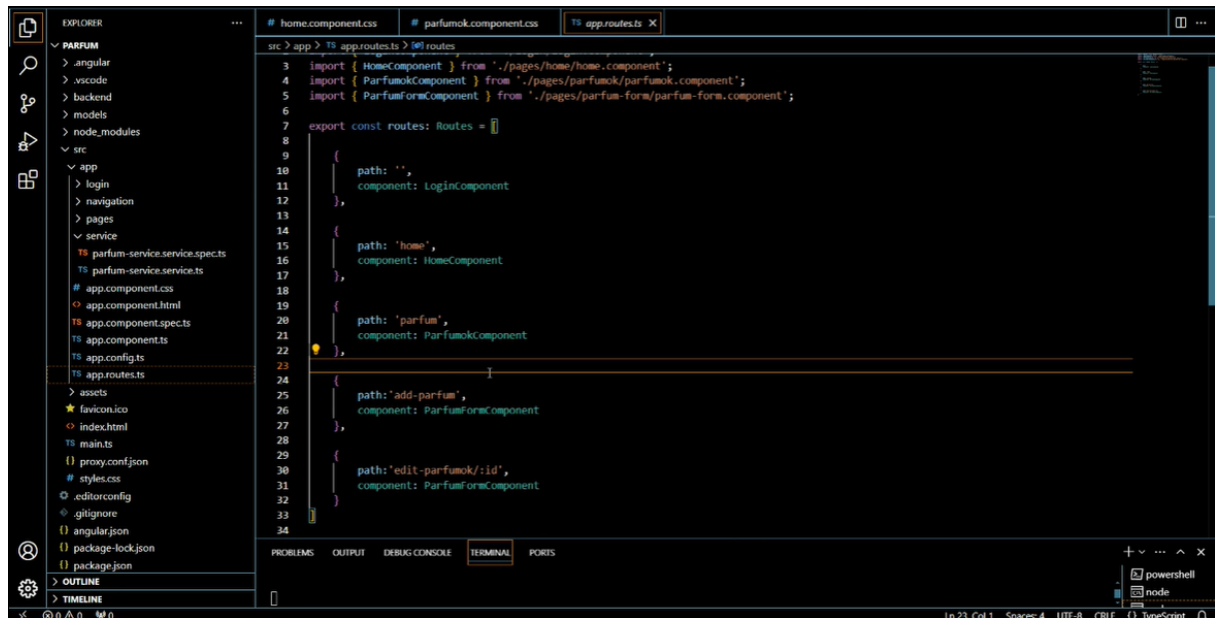
```
1 <div class="row">
2   <div class="col md 12">
3     <table class="table table-striped">
4       <thead>
5         <tr>
6           <th>ID</th>
7           <th>Name</th>
8           <th>Brand</th>
9           <th>Price</th>
10          <th>Quantity</th>
11          <th>Actions</th>
12        </tr>
13      </thead>
14      <tbody>
15        @for(parfum of parfums; track $index) {
16          <tr>
17            <td>{{ parfum.id }}</td>
18            <td>{{ parfum.name }}</td>
19            <td>{{ parfum.brand }}</td>
20            <td>{{ parfum.price }}</td>
21            <td>{{ parfum.quantity }}</td>
22            <td>
23              <button class="btn btn-outline-primary" (click)="goToUserForm(parfum.id)">
24                Modify
25              </button>
26              <button class="btn btn-outline-danger ms-2" (click)="deleteUser(parfum)">
27                Delete
28              </button>
29            </td>
30          </tr>
31        }
32      </tbody>
33    </table>
34  </div>
35 </div>
```


Service:



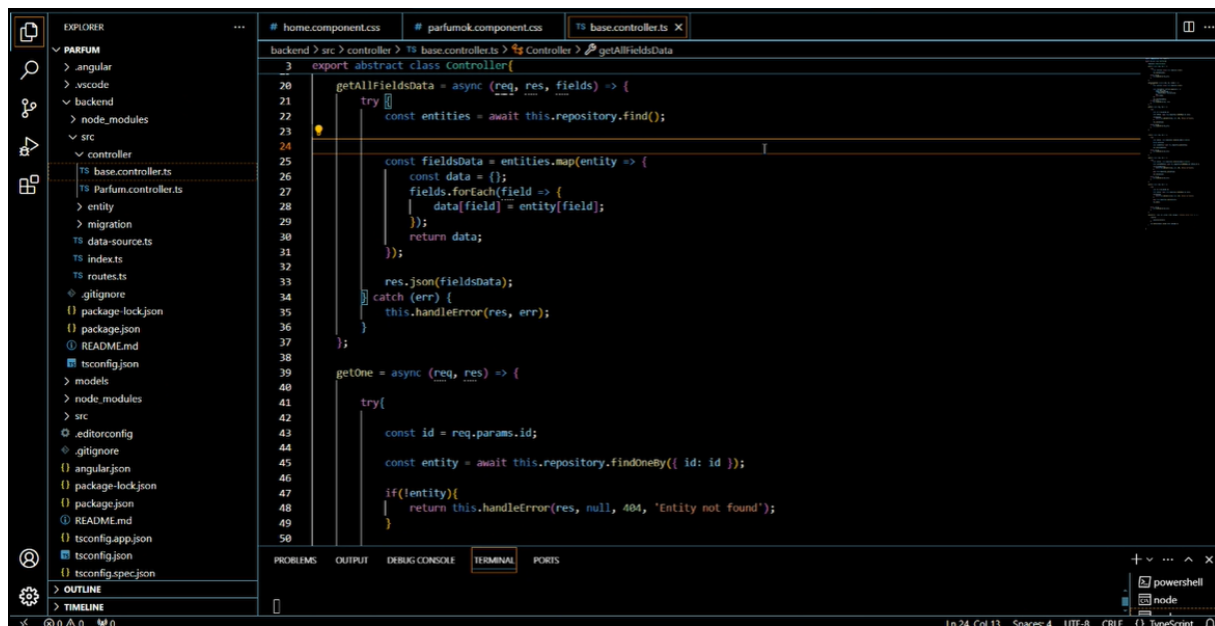
```
6  })
7  export class ParfumsService {
8
9      http = inject(HttpClient);
10
11      getAll() {
12          return this.http.get<ParfumDTO>('/api/parfum');
13      }
14
15      getAllFieldsData() {
16          return this.http.get<ParfumDTO>('/api/parfum/id');
17      }
18
19      getOne(id: number) {
20          return this.http.get<ParfumDTO>('/api/parfum/' + id);
21      }
22
23      create(parfum: ParfumDTO) {
24          return this.http.post<ParfumDTO>('/api/parfum', parfum);
25      }
26
27      update(parfum: ParfumDTO) {
28          return this.http.put<ParfumDTO>('/api/parfum', parfum);
29      }
30
31      delete(id: number) {
32          return this.http.delete('/api/parfum/' + id);
33      }
34  }
35
```

Route:



```
3  import { HomeComponent } from './pages/home/home.component';
4  import { ParfumsComponent } from './pages/parfums/parfums.component';
5  import { ParfumFormComponent } from './pages/parfum-form/parfum-form.component';
6
7  export const routes: Routes = [
8
9      {
10         path: '',
11         component: LoginComponent
12     },
13
14     {
15         path: 'home',
16         component: HomeComponent
17     },
18
19     {
20         path: 'parfum',
21         component: ParfumsComponent
22     },
23
24     {
25         path: 'add-parfum',
26         component: ParfumFormComponent
27     },
28
29     {
30         path: 'edit-parfum/:id',
31         component: ParfumFormComponent
32     },
33 ]
```

Controller:



The screenshot shows a Visual Studio Code editor with a project named 'PARFUM'. The Explorer sidebar on the left shows the file structure, with 'src' expanded to show 'controller'. The file 'base.controller.ts' is selected. The main editor displays the code for this file, which defines an abstract class 'Controller' with two methods: 'getAllFieldsData' and 'getOne'. The 'getAllFieldsData' method is an asynchronous function that takes 'req', 'res', and 'fields' as arguments. It uses 'this.repository.find()' to fetch entities, maps them to a 'fieldsData' object, and returns the data as JSON. The 'getOne' method is also asynchronous, taking 'req' and 'res' as arguments. It uses 'this.repository.findOneBy()' to find an entity by ID and returns an error if the entity is not found. The bottom of the editor shows the 'TERMINAL' tab with a 'powershell' prompt.

```
3 export abstract class Controller {
20   getAllFieldsData = async (req, res, fields) => {
21     try {
22       const entities = await this.repository.find();
23
24       const fieldsData = entities.map(entity => {
25         const data = {};
26         fields.forEach(field => {
27           data[field] = entity[field];
28         });
29         return data;
30       });
31       res.json(fieldsData);
32     } catch (err) {
33       this.handleError(res, err);
34     }
35   };
36
37   getOne = async (req, res) => {
38     try {
39       const id = req.params.id;
40
41       const entity = await this.repository.findOneBy({ id: id });
42
43       if (!entity) {
44         return this.handleError(res, null, 404, 'Entity not found');
45       }
46     }
47   };
48 }
49
50
```