SAVAT

The cheapest prices only in SAVAT

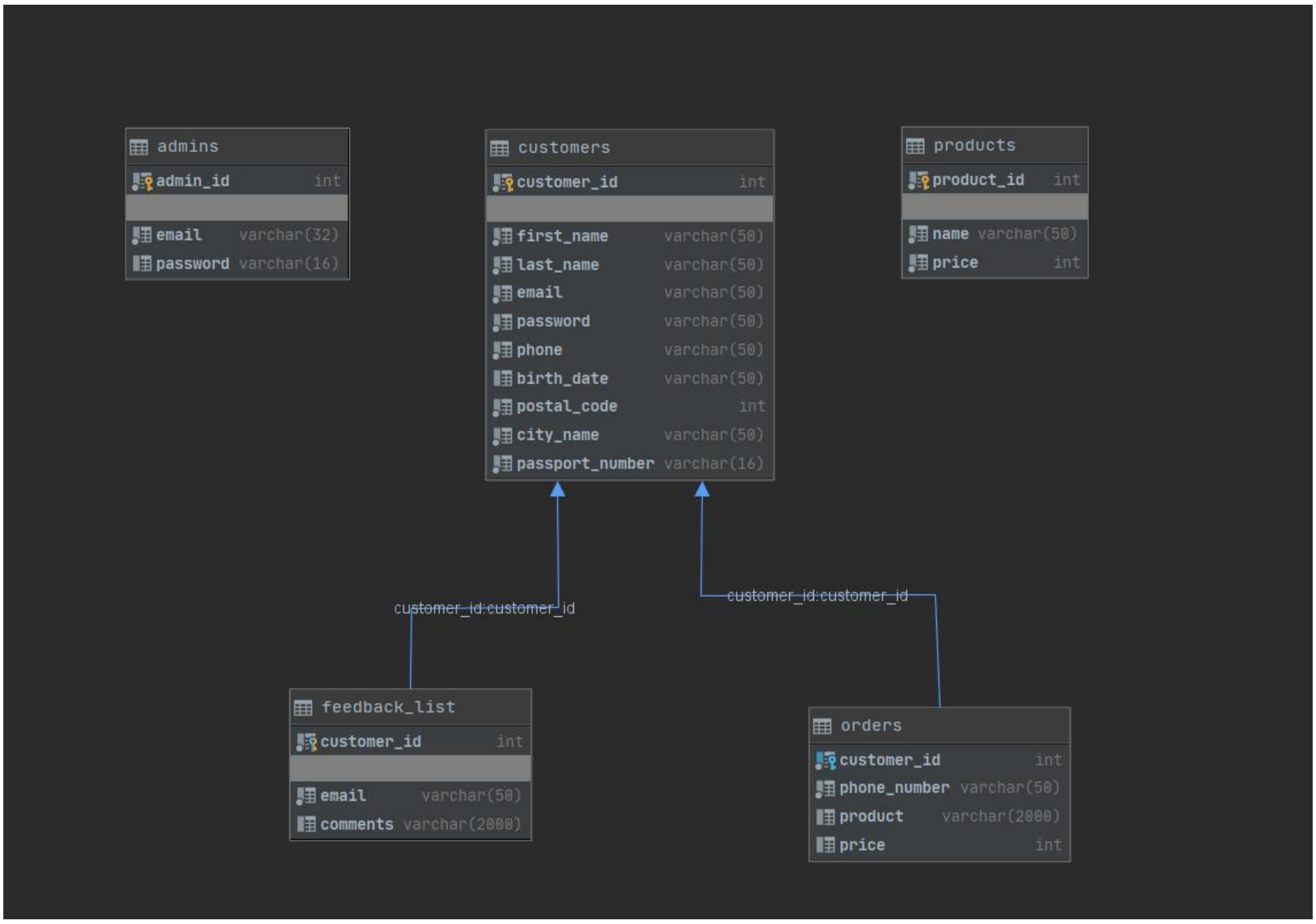
Team Members:
Abdukhamidov Saidazim -
U1810210 Fozilov Bobur - U1810105
Akhmedova Ziyoda - U1810189
Kenjaeva Mekhriniso - U1810285
Tashpulatova Azizabonu - U1810286

GitHub repository:
https://github.com/iuthub/ip-group-project-sophomores-online-store/tr ee/master/savat

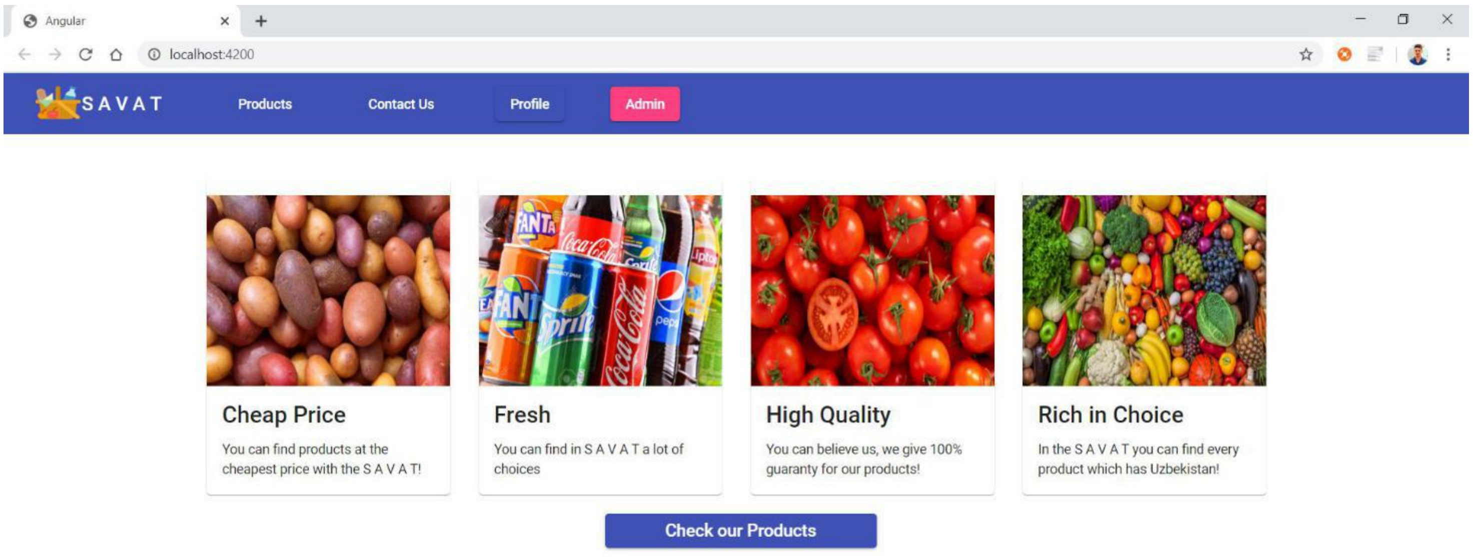
SAVAT is an online market for only eatables as drinks and vegetables. This web application can help people to find food and order them. In this application, we have 2 roles Admins and Customers. Admins can add **products**, manage **orders**, and **feedback**. Customers can search for **products** in our web application and **order** them by inserting just their phone number and send **feedback** to SAVAT managers.

We used **Angular** for making Single Page Application to make experience of user easier and faster. Also, we used **Angular Material** for styled components and **Bootstrap** for making our application responsive. **PHP** is used for only connecting to database and getting data using **REST API**s.

We created 5 tables :
Customers, admins, products, feedback_list and orders.
All registered customers are inserted into **customers** table. Admins are in the **admins** table. **Produts** table is for saving products. **Feedback_list** and **orders** tables have relationship with **customers** table their keys are connected to customer_id in the **customers** table.



When user starts using this application, firstly landing page will be opened.



Cheap Price

You can find products at the cheapest price with the S A V A T!

Fresh

You can find in S A V A T a lot of choices

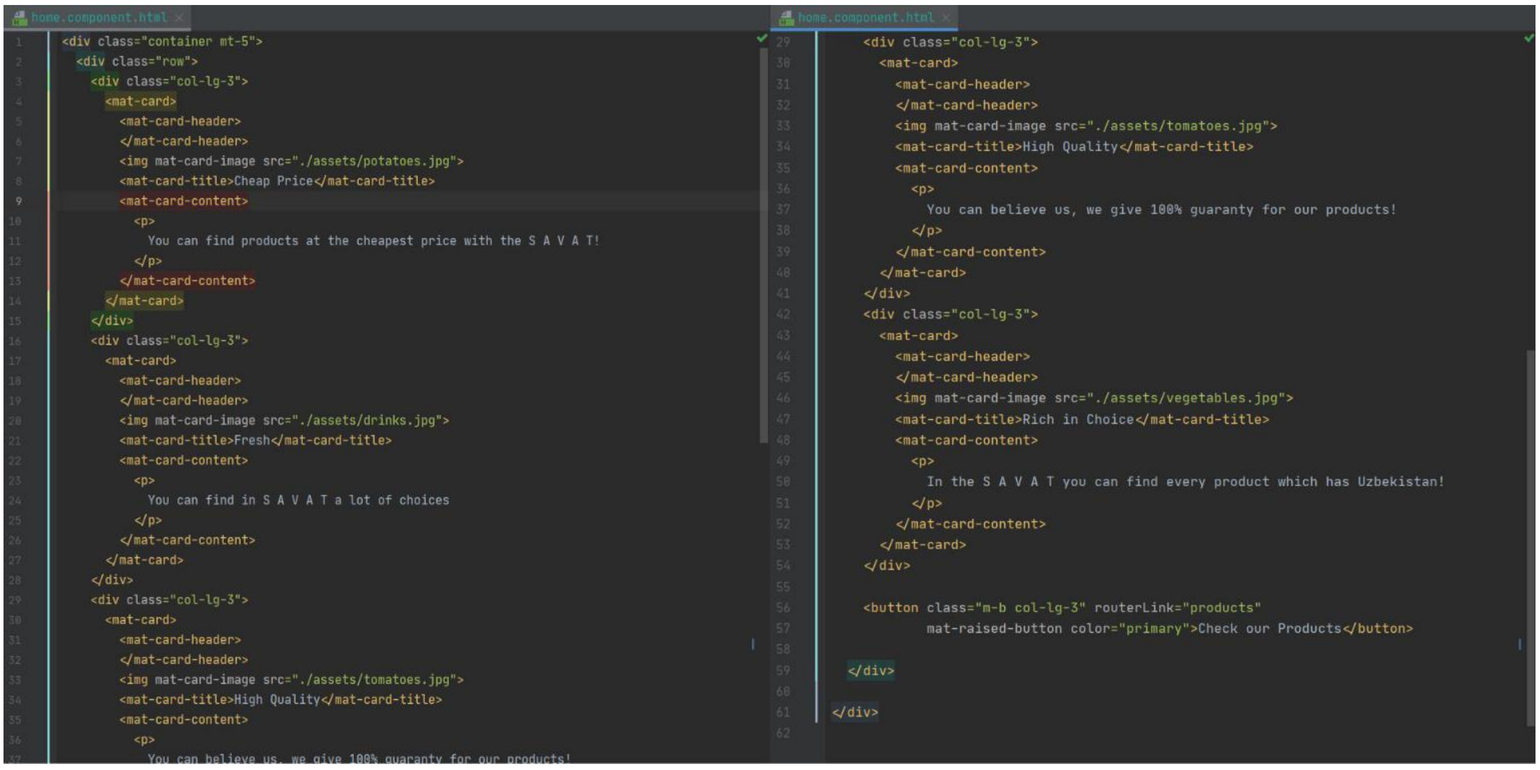
High Quality

You can believe us, we give 100% guaranty for our products!

Rich in Choice

In the S A V A T you can find every product which has Uzbekistan!

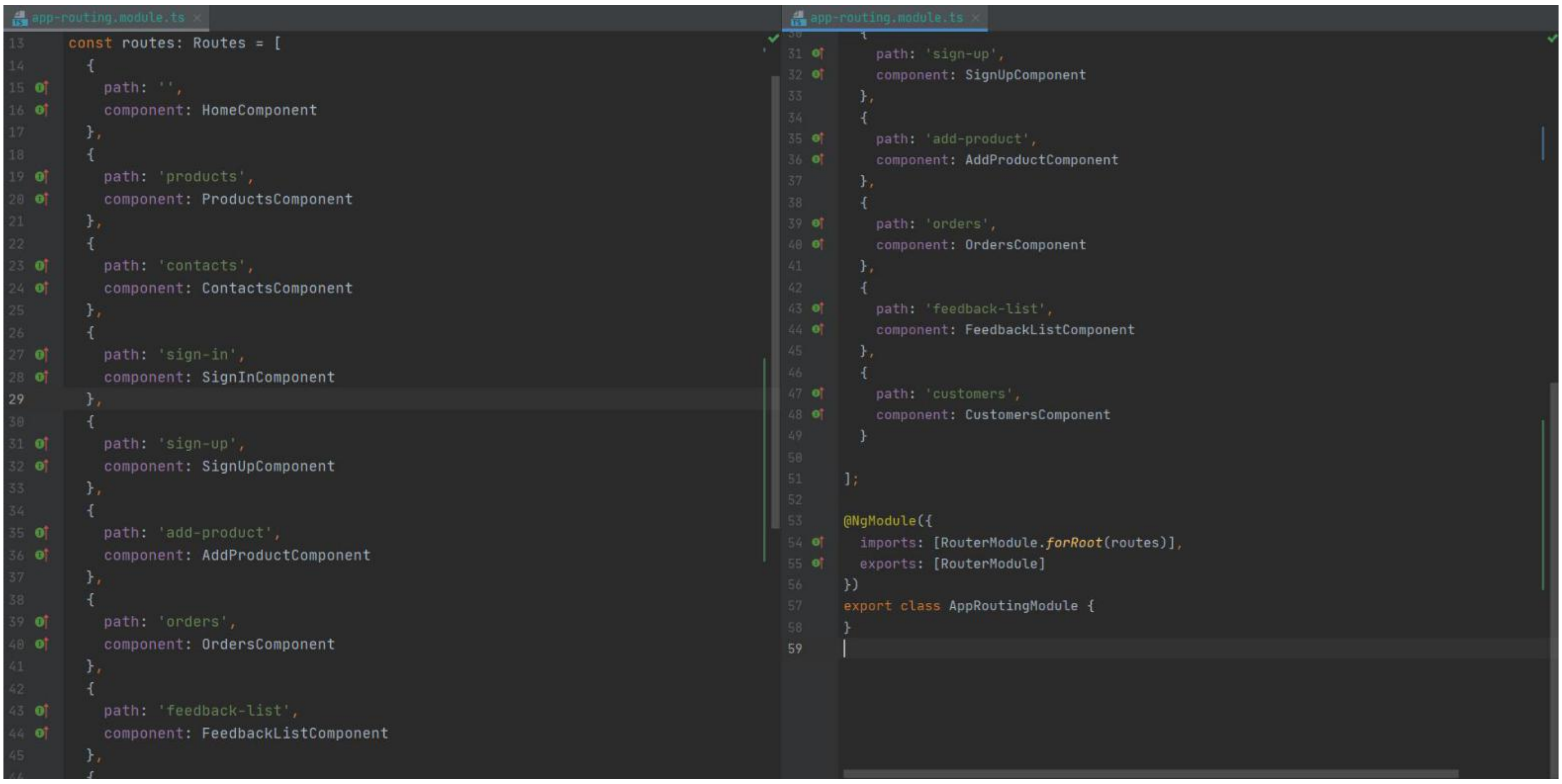
In *angular/src/app/components/home* directory there is home component for implementing this page.



```html
<div class="container mt-5">
  <div class="row">
    <div class="col-lg-3">
      <mat-card>
        <mat-card-header>
        </mat-card-header>
        <img mat-card-image src="./assets/potatoes.jpg">
        <mat-card-title>Cheap Price</mat-card-title>
        <mat-card-content>
          <p>
            You can find products at the cheapest price with the S A V A T!
          </p>
        </mat-card-content>
      </mat-card>
    </div>
    <div class="col-lg-3">
      <mat-card>
        <mat-card-header>
        </mat-card-header>
        <img mat-card-image src="./assets/drinks.jpg">
        <mat-card-title>Fresh</mat-card-title>
        <mat-card-content>
          <p>
            You can find in S A V A T a lot of choices
          </p>
        </mat-card-content>
      </mat-card>
    </div>
    <div class="col-lg-3">
      <mat-card>
        <mat-card-header>
        </mat-card-header>
        <img mat-card-image src="./assets/tomatoes.jpg">
        <mat-card-title>High Quality</mat-card-title>
        <mat-card-content>
          <p>
            You can believe us, we give 100% guaranty for our products!
```

```html
    <div class="col-lg-3">
      <mat-card>
        <mat-card-header>
        </mat-card-header>
        <img mat-card-image src="./assets/tomatoes.jpg">
        <mat-card-title>High Quality</mat-card-title>
        <mat-card-content>
          <p>
            You can believe us, we give 100% guaranty for our products!
          </p>
        </mat-card-content>
      </mat-card>
    </div>
    <div class="col-lg-3">
      <mat-card>
        <mat-card-header>
        </mat-card-header>
        <img mat-card-image src="./assets/vegetables.jpg">
        <mat-card-title>Rich in Choice</mat-card-title>
        <mat-card-content>
          <p>
            In the S A V A T you can find every product which has Uzbekistan!
          </p>
        </mat-card-content>
      </mat-card>
    </div>

    <button class="m-b col-lg-3" routerLink="products"
            mat-raised-button color="primary">Check our Products</button>

  </div>

</div>
```

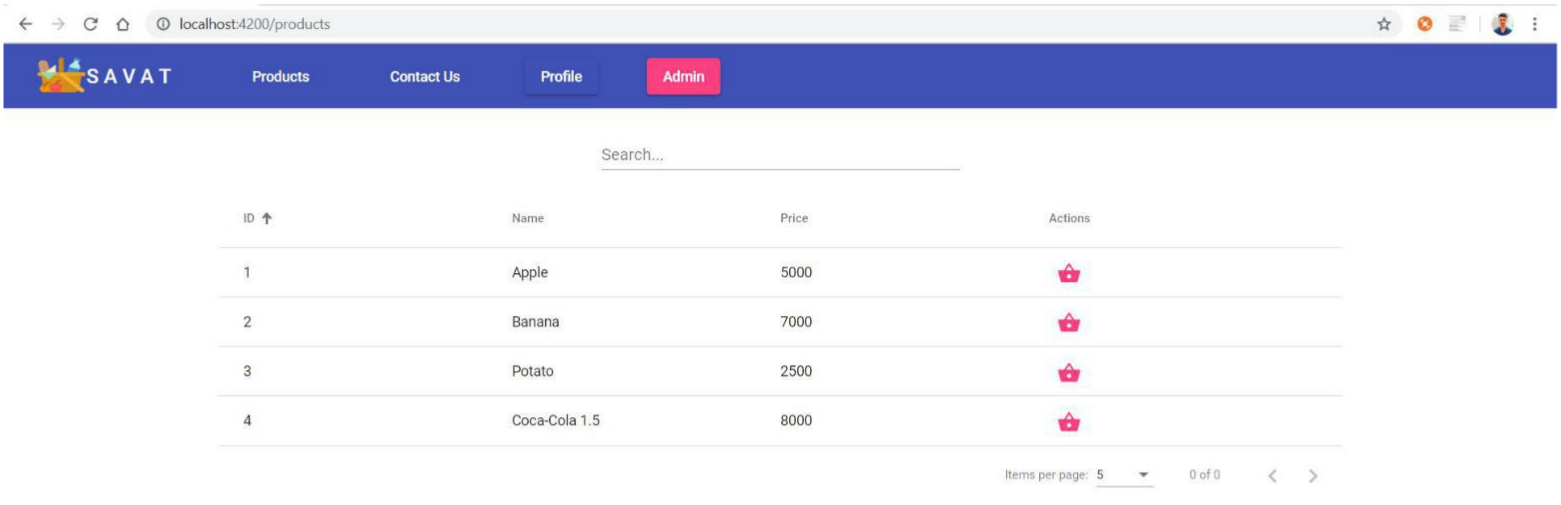
This is our *app-routing.module* All our routerLinks implemented in this file.



```
const routes: Routes = [
  {
    path: '',
    component: HomeComponent
  },
  {
    path: 'products',
    component: ProductsComponent
  },
  {
    path: 'contacts',
    component: ContactsComponent
  },
  {
    path: 'sign-in',
    component: SignInComponent
  },
  {
    path: 'sign-up',
    component: SignUpComponent
  },
  {
    path: 'add-product',
    component: AddProductComponent
  },
  {
    path: 'orders',
    component: OrdersComponent
  },
  {
    path: 'feedback-list',
    component: FeedbackListComponent
  },
```

```
    path: 'sign-up',
    component: SignUpComponent
  },
  {
    path: 'add-product',
    component: AddProductComponent
  },
  {
    path: 'orders',
    component: OrdersComponent
  },
  {
    path: 'feedback-list',
    component: FeedbackListComponent
  },
  {
    path: 'customers',
    component: CustomersComponent
  }

];

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

If user navigate to Products, products list will be opened



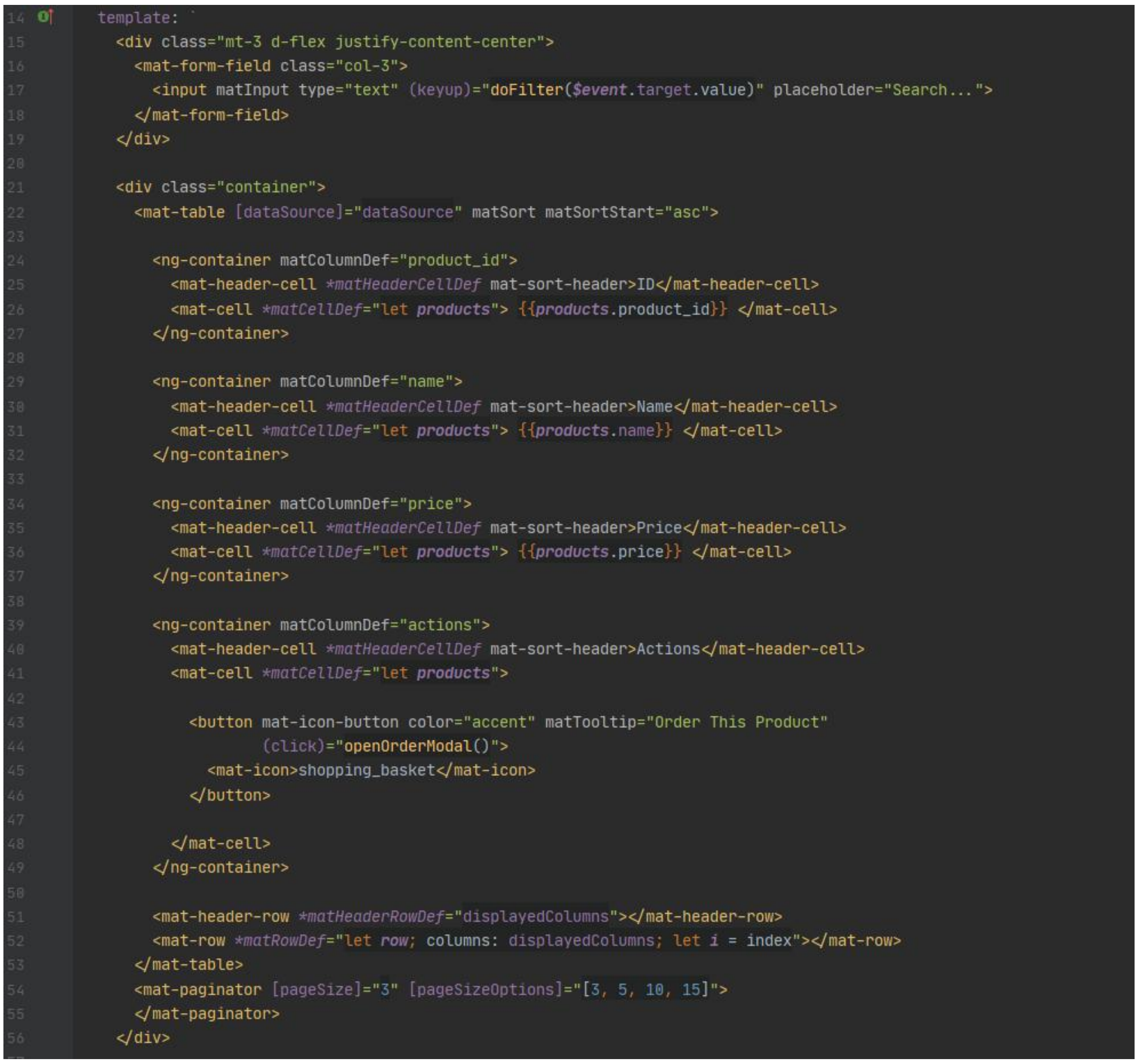
ID	Name	Price	Actions
1	Apple	5000	
2	Banana	7000	
3	Potato	2500	
4	Coca-Cola 1.5	8000	

Items per page: 5   0 of 0

We used Table of Angular Material, in this table we can do **searching, sorting, paging.**

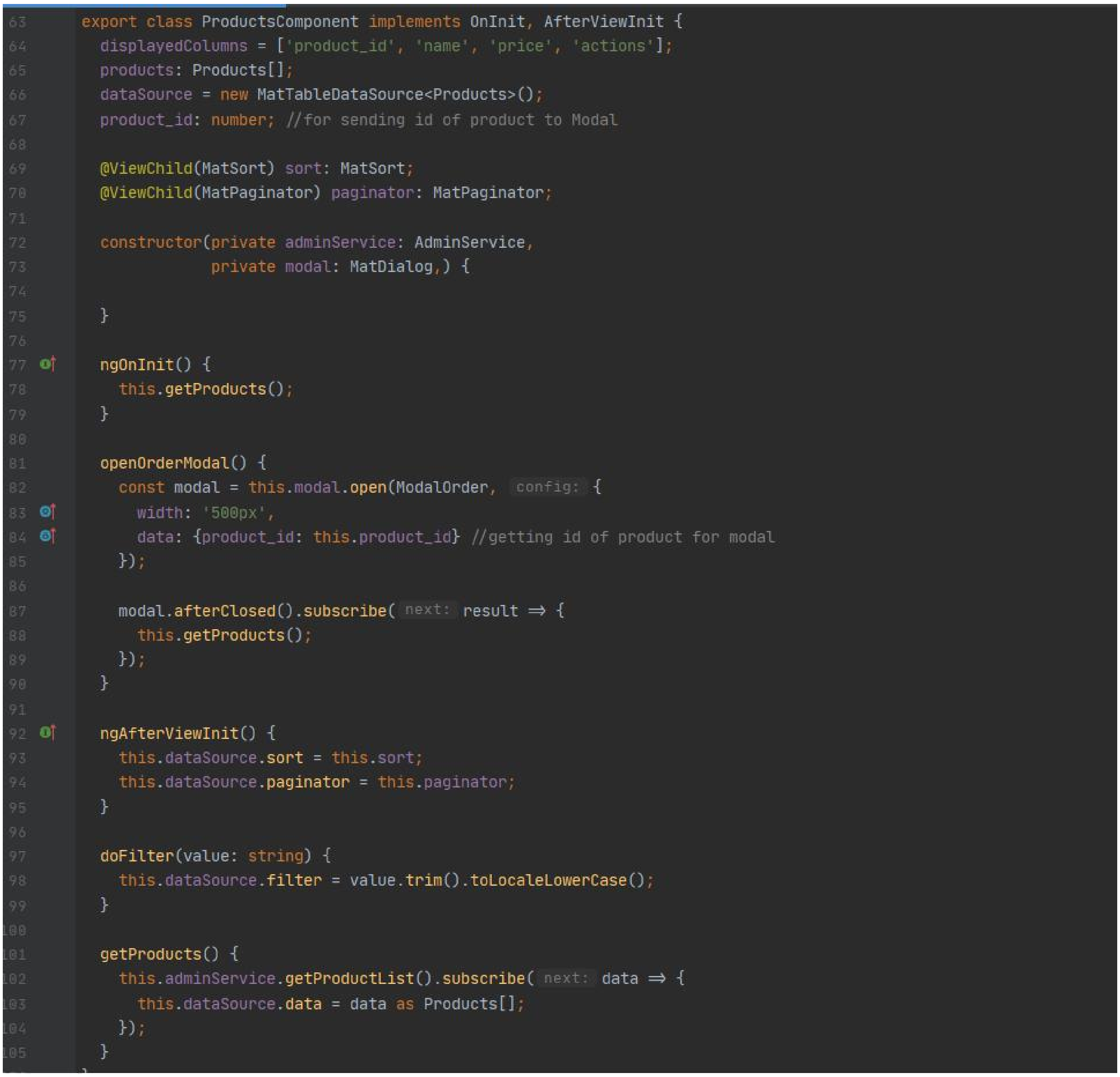
Implementation of this page in: *app/components/products/products.ts*

This is template for Products page



```
template: `
  <div class="mt-3 d-flex justify-content-center">
    <mat-form-field class="col-3">
      <input matInput type="text" (keyup)="doFilter($event.target.value)" placeholder="Search...">
    </mat-form-field>
  </div>

  <div class="container">
    <mat-table [dataSource]="dataSource" matSort matSortStart="asc">

      <ng-container matColumnDef="product_id">
        <mat-header-cell *matHeaderCellDef mat-sort-header>ID</mat-header-cell>
        <mat-cell *matCellDef="let products"> {{products.product_id}} </mat-cell>
      </ng-container>

      <ng-container matColumnDef="name">
        <mat-header-cell *matHeaderCellDef mat-sort-header>Name</mat-header-cell>
        <mat-cell *matCellDef="let products"> {{products.name}} </mat-cell>
      </ng-container>

      <ng-container matColumnDef="price">
        <mat-header-cell *matHeaderCellDef mat-sort-header>Price</mat-header-cell>
        <mat-cell *matCellDef="let products"> {{products.price}} </mat-cell>
      </ng-container>

      <ng-container matColumnDef="actions">
        <mat-header-cell *matHeaderCellDef mat-sort-header>Actions</mat-header-cell>
        <mat-cell *matCellDef="let products">

          <button mat-icon-button color="accent" matTooltip="Order This Product"
                  (click)="openOrderModal()">
            <mat-icon>shopping_basket</mat-icon>
          </button>

        </mat-cell>
      </ng-container>

      <mat-header-row *matHeaderRowDef="displayedColumns"></mat-header-row>
      <mat-row *matRowDef="let row; columns: displayedColumns; let i = index"></mat-row>
    </mat-table>
    <mat-paginator [pageSize]="3" [pageSizeOptions]="[3, 5, 10, 15]">
    </mat-paginator>
  </div>
```

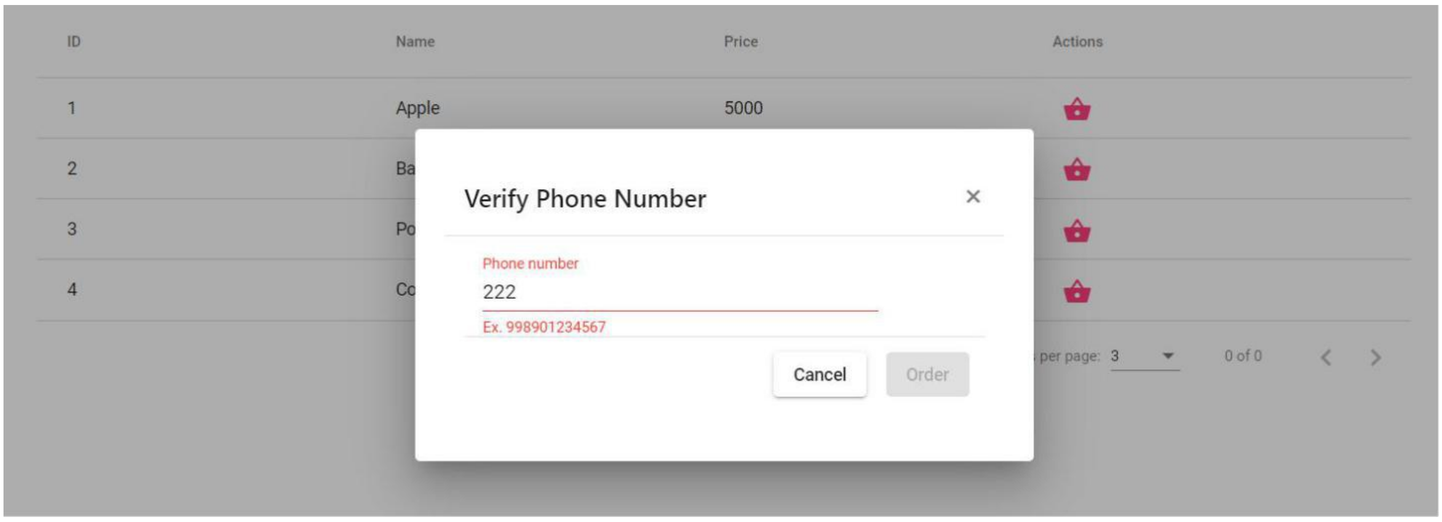
This is logic of **Producs** page. In the AdminService there are all functions for manipulationg with **REST API** using **HttpClient**.



```
export class ProductsComponent implements OnInit, AfterViewInit {
  displayedColumns = ['product_id', 'name', 'price', 'actions'];
  products: Products[];
  dataSource = new MatTableDataSource<Products>();
  product_id: number; //for sending id of product to Modal

  @ViewChild(MatSort) sort: MatSort;
  @ViewChild(MatPaginator) paginator: MatPaginator;

  constructor(private adminService: AdminService,
              private modal: MatDialog,) {

  }

  ngOnInit() {
    this.getProducts();
  }

  openOrderModal() {
    const modal = this.modal.open(ModalOrder,  config: {
      width: '500px',
      data: {product_id: this.product_id} //getting id of product for modal
    });

    modal.afterClosed().subscribe( next: result => {
      this.getProducts();
    });
  }

  ngAfterViewInit() {
    this.dataSource.sort = this.sort;
    this.dataSource.paginator = this.paginator;
  }

  doFilter(value: string) {
    this.dataSource.filter = value.trim().toLocaleLowerCase();
  }

  getProducts() {
    this.adminService.getProductList().subscribe( next: data => {
      this.dataSource.data = data as Products[];
    });
  }
```

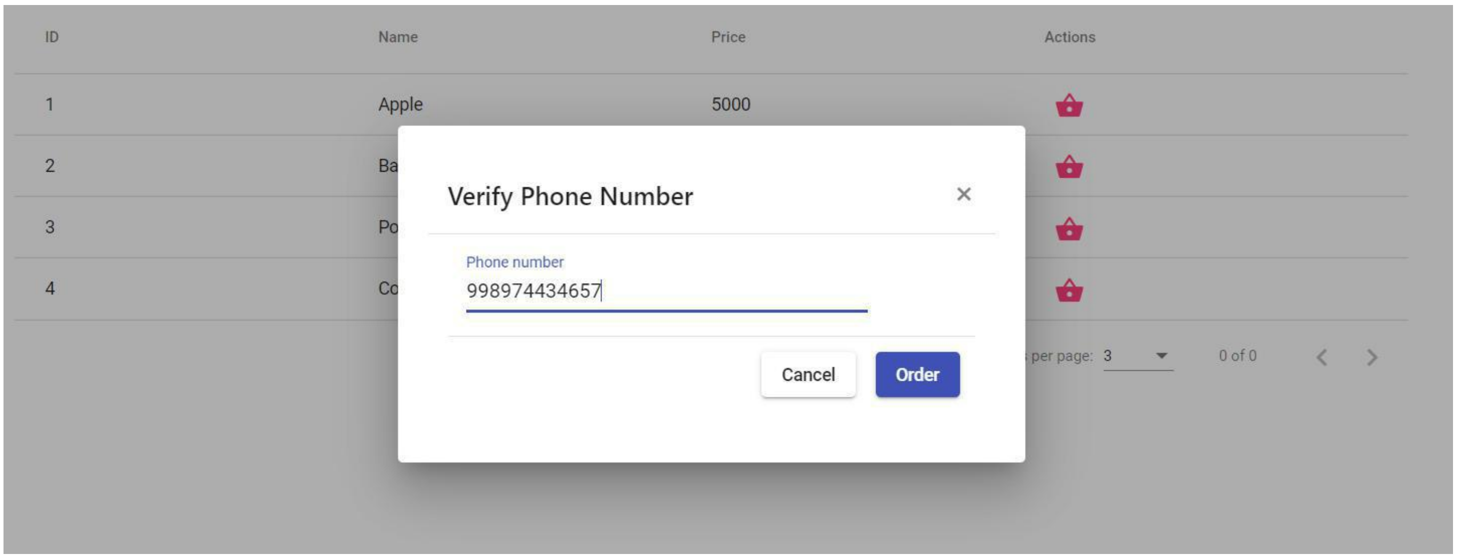
openOrderModal() function is for opening modal for ordering product. That means:

If we click to **basket** icon:



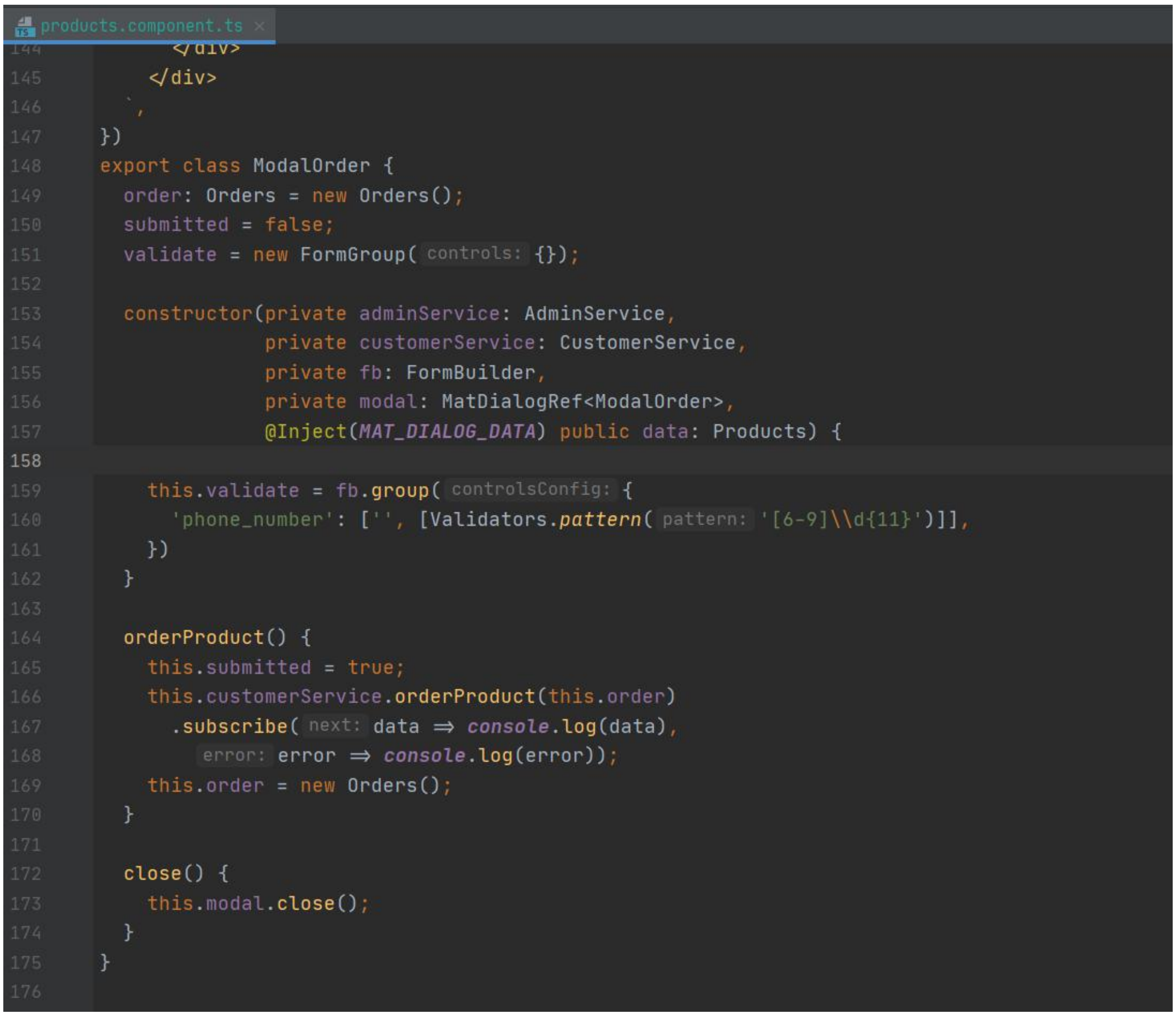
User should be insert own phone number and we used **validation** for escaping mistakes. If there is mistake, **Order** button will be **disabled.**

After writing correctly:



Errors will disappear and **Order** button will be usable.

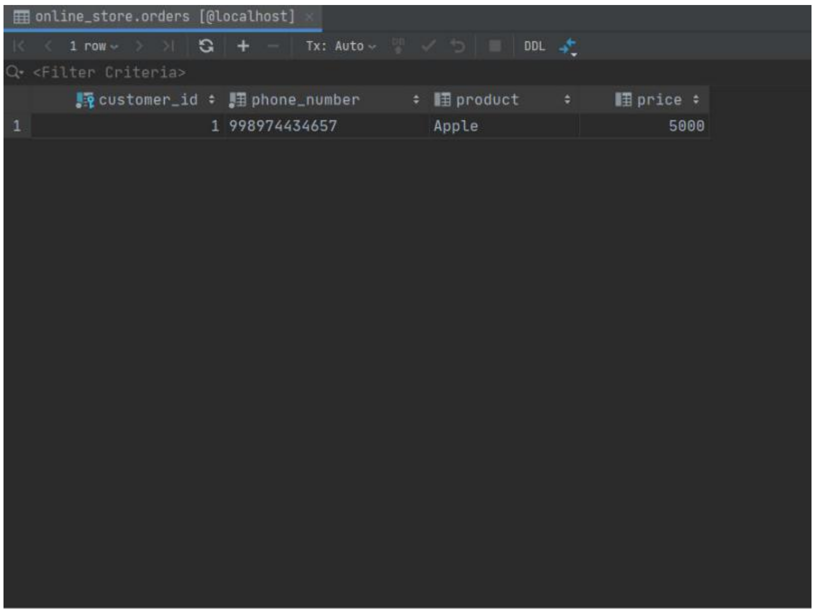
We used Angular’s **FormBuilder** validator **patterns** for validate that input.



products.component.ts

```
        </div>
      </div>
    `,
})
export class ModalOrder {
  order: Orders = new Orders();
  submitted = false;
  validate = new FormGroup( controls: {});

  constructor(private adminService: AdminService,
              private customerService: CustomerService,
              private fb: FormBuilder,
              private modal: MatDialogRef<ModalOrder>,
              @Inject(MAT_DIALOG_DATA) public data: Products) {

    this.validate = fb.group( controlsConfig: {
      'phone_number': ['', [Validators.pattern( pattern: '[6-9]\\d{11}')]],
    })
  }

  orderProduct() {
    this.submitted = true;
    this.customerService.orderProduct(this.order)
      .subscribe( next: data => console.log(data),
        error: error => console.log(error));
    this.order = new Orders();
  }

  close() {
    this.modal.close();
  }
}
```

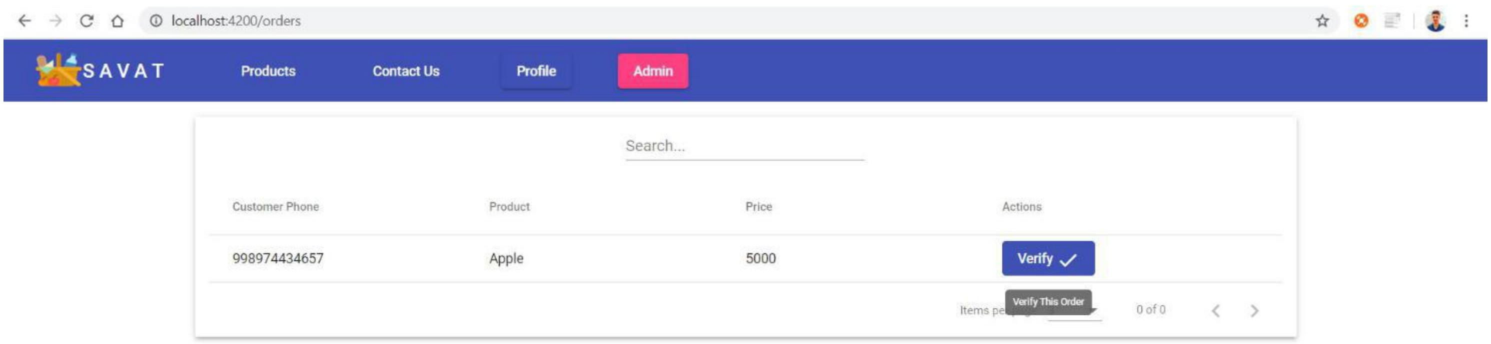
After ordering correctly, order will appear in **orders** table:


online_store.orders [@localhost]

<Filter Criteria>

	customer_id	phone_number	product	price
1	1	998974434657	Apple	5000

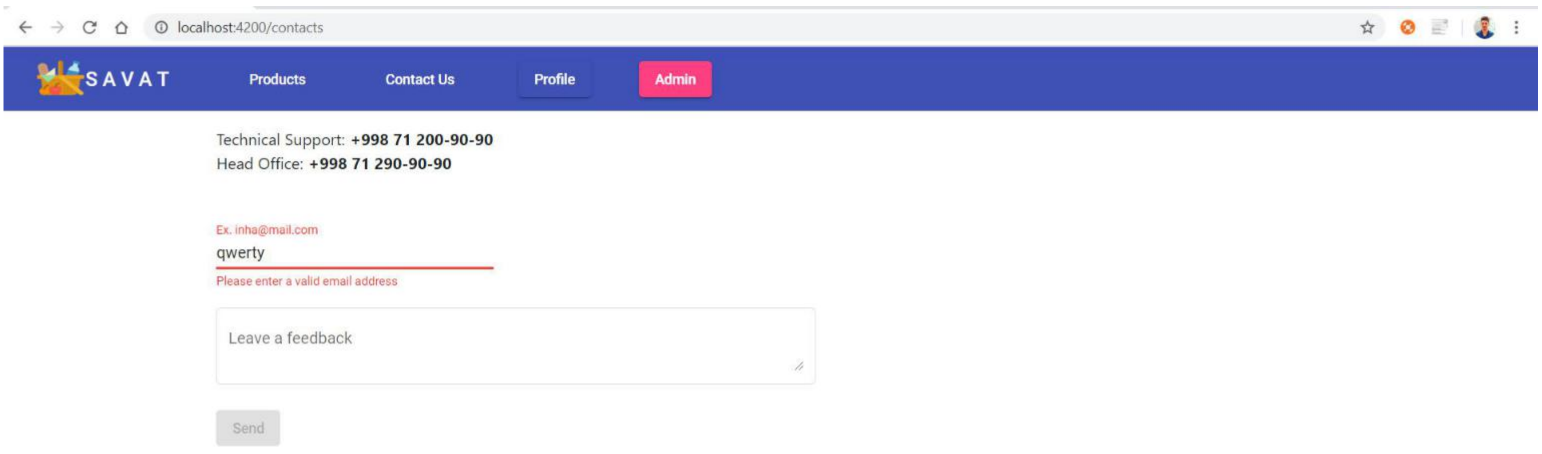
Admin can see and **verify** that order in **Orders List.**


localhost:4200/orders

Customer Phone	Product	Price	Actions
998974434657	Apple	5000	Verify ✓

Also, users can send **feedback** to managers using **Contact Us** link:

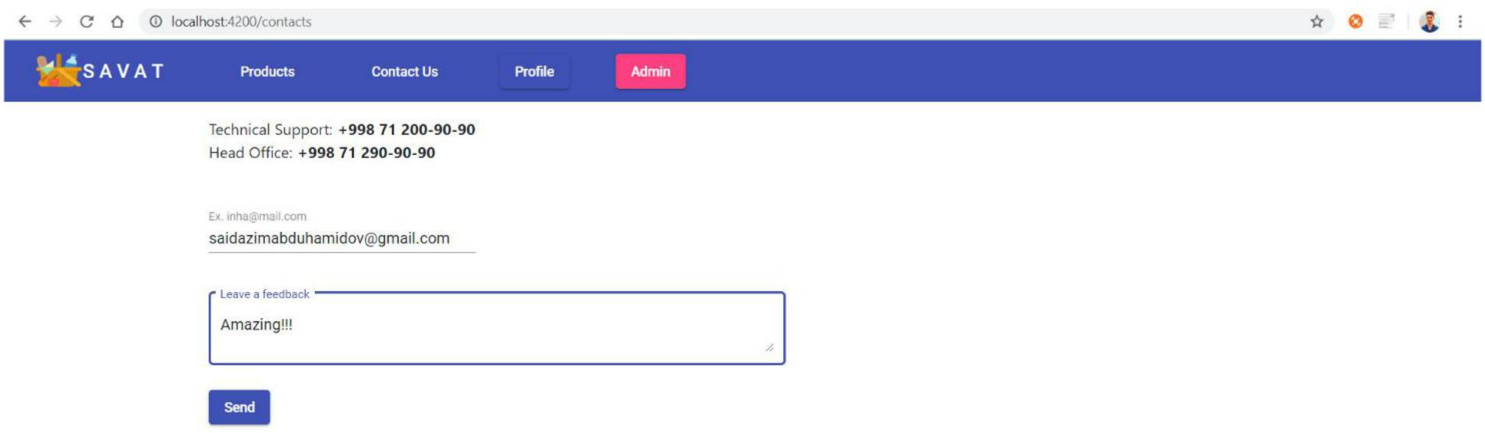
If user insert invalid email address error appears instantly, and **Send** button will be disabled


localhost:4200/contacts

Technical Support: **+998 71 200-90-90**
Head Office: **+998 71 290-90-90**

Ex. inha@mail.com
qwerty
Please enter a valid email address

Leave a feedback

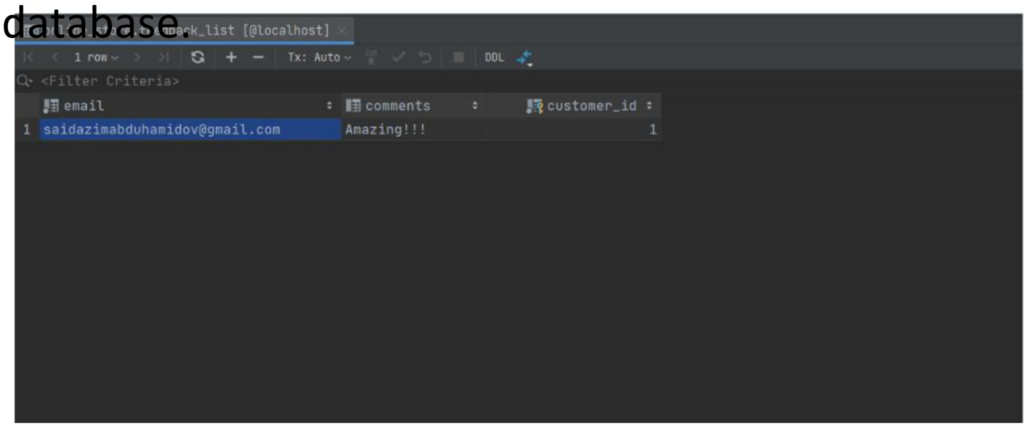

localhost:4200/contacts

Technical Support: **+998 71 200-90-90**
Head Office: **+998 71 290-90-90**

Ex. inha@mail.com
saidazimabduhamidov@gmail.com

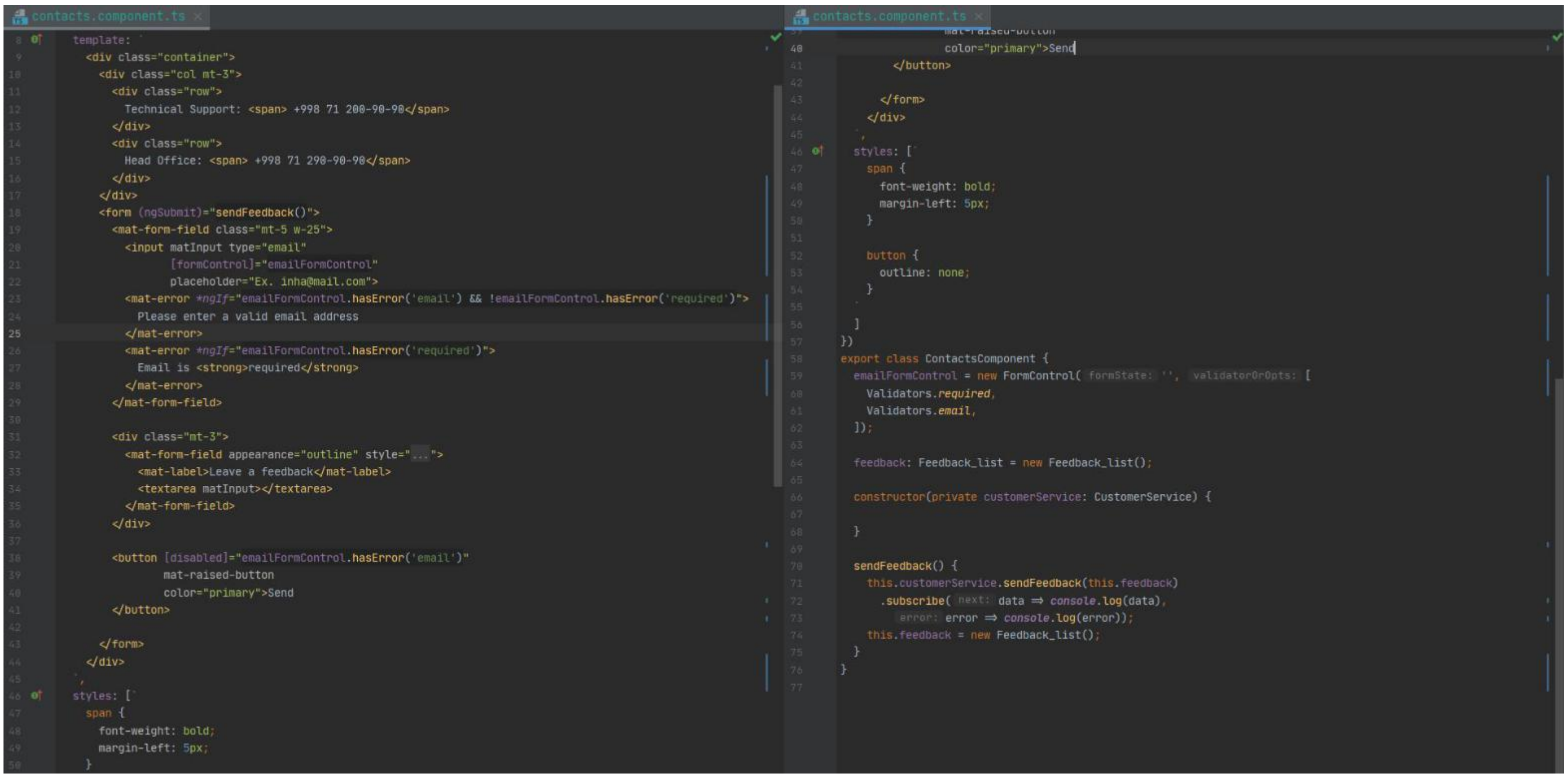
Leave a feedback
Amazing!!!

After inserting values correctly, **feedback** will be inserted to the **feedback_list** table in database.


online_store.feedback_list [@localhost]

	email	comments	customer_id
1	saidazimabduhamidov@gmail.com	Amazing!!!	1

In **sendFeedback()** function we used sendFeedback function of **CustomerService** for **POST** HTTP request



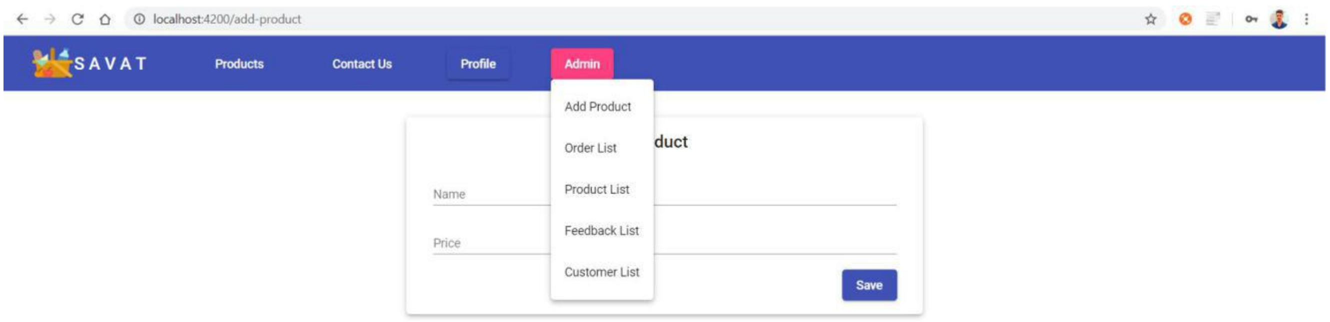
contacts.component.ts

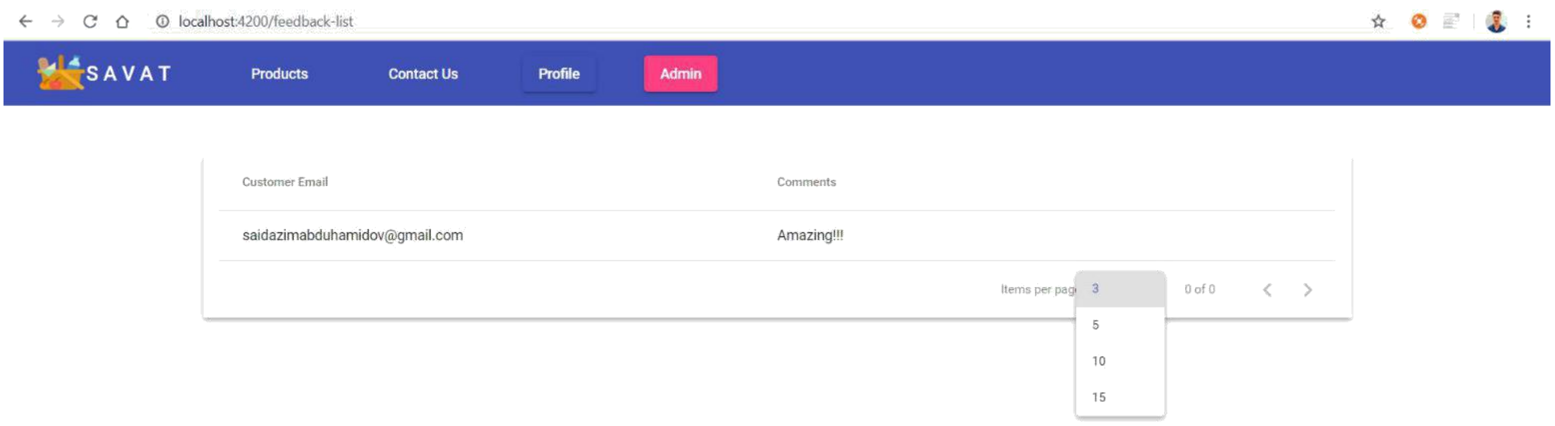
```
  template: `
    <div class="container">
      <div class="col mt-3">
        <div class="row">
          Technical Support: <span> +998 71 200-90-90</span>
        </div>
        <div class="row">
          Head Office: <span> +998 71 290-90-90</span>
        </div>
      </div>
      <form (ngSubmit)="sendFeedback()">
        <mat-form-field class="mt-5 w-25">
          <input matInput type="email"
                 [formControl]="emailFormControl"
                 placeholder="Ex. inha@mail.com">
          <mat-error *ngIf="emailFormControl.hasError('email') && !emailFormControl.hasError('required')">
            Please enter a valid email address
          </mat-error>
          <mat-error *ngIf="emailFormControl.hasError('required')">
            Email is <strong>required</strong>
          </mat-error>
        </mat-form-field>

        <div class="mt-3">
          <mat-form-field appearance="outline" style="...">
            <mat-label>Leave a feedback</mat-label>
            <textarea matInput></textarea>
          </mat-form-field>
        </div>

        <button [disabled]="emailFormControl.hasError('email')"
                mat-raised-button
                color="primary">Send
        </button>

      </form>
    </div>
    `,
  styles: [`
    span {
      font-weight: bold;
      margin-left: 5px;
    }
```

contacts.component.ts

```
                mat-raised-button
                color="primary">Send
        </button>

      </form>
    </div>
    `,
  styles: [`
    span {
      font-weight: bold;
      margin-left: 5px;
    }

    button {
      outline: none;
    }
    `
  ]
})
export class ContactsComponent {
  emailFormControl = new FormControl( formState: '', validatorOrOpts: [
    Validators.required,
    Validators.email,
  ]);

  feedback: Feedback_list = new Feedback_list();

  constructor(private customerService: CustomerService) {

  }

  sendFeedback() {
    this.customerService.sendFeedback(this.feedback)
      .subscribe( next: data => console.log(data),
        error: error => console.log(error));
    this.feedback = new Feedback_list();
  }
}
```

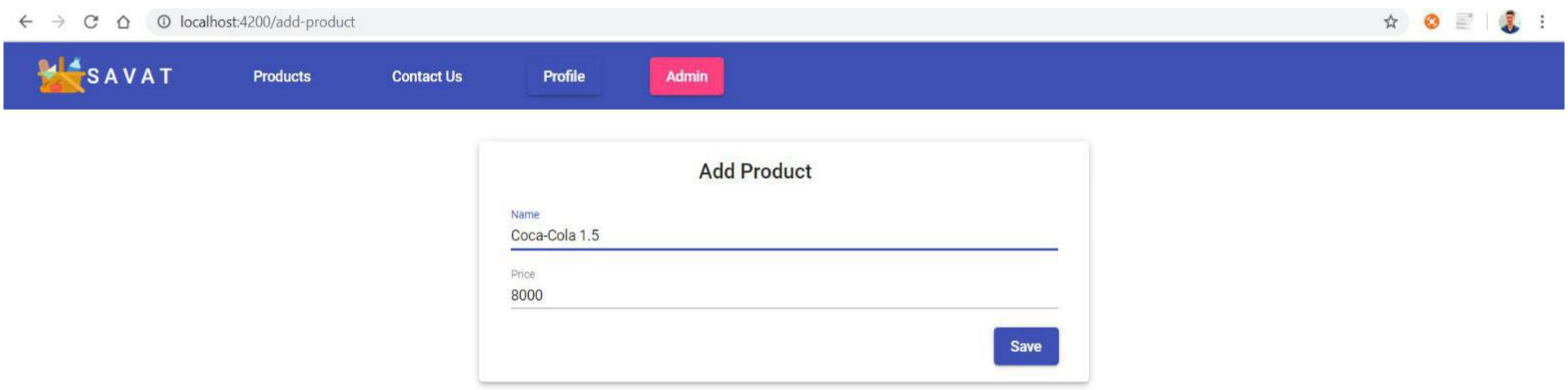
After putting feedback in database, Admins can see **feedback** in **Feedback List** link:





Customer Email	Comments
saidazimabduhamidov@gmail.com	Amazing!!!

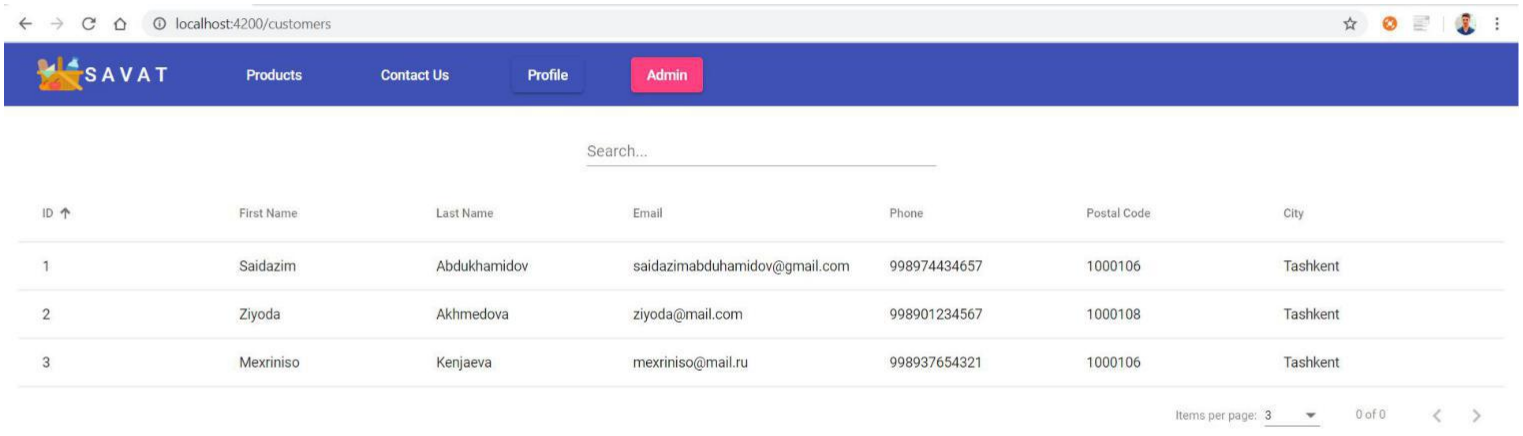
If user click **Add Product** button in the menu of **Admin** button, user can add product by this form



Name
Coca-Cola 1.5

Price
8000

And **Admin** can see list of registered customers:

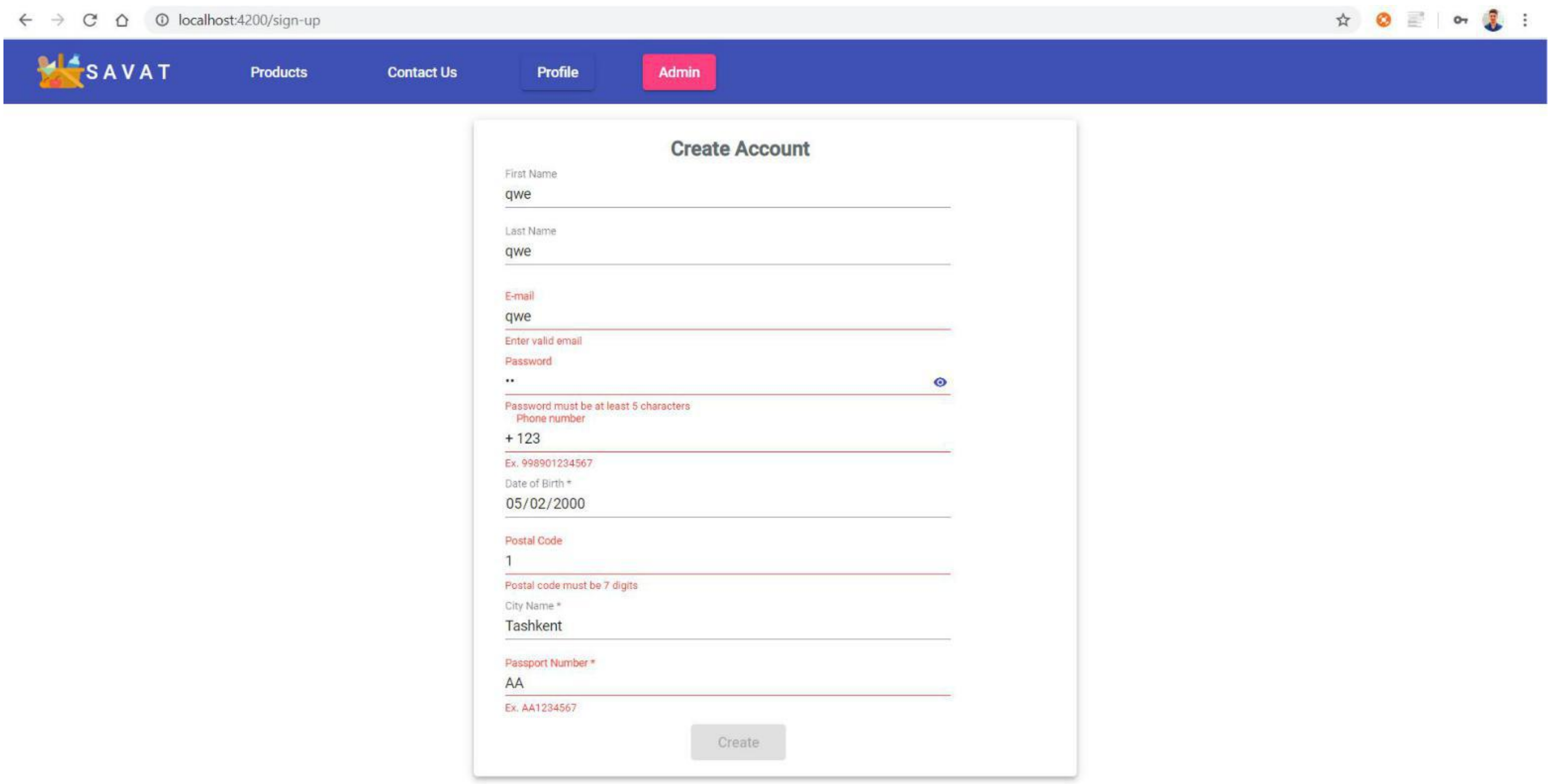


ID	First Name	Last Name	Email	Phone	Postal Code	City
1	Saidazim	Abdukhamidov	saidazimabduhamidov@gmail.com	998974434657	1000106	Tashkent
2	Ziyoda	Akhmedova	ziyoda@mail.com	998901234567	1000108	Tashkent
3	Mexriniso	Kenjaeva	mexriniso@mail.ru	998937654321	1000106	Tashkent

Items per page: 3 0 of 0

Finally, **Login** and **Registration** pages. Users can Login or make Registration using **Profile** button in the **nav-bar**

In the registration page, we put validations for important values. For example:



Create Account

First Name
qwe

Last Name
qwe

E-mail
qwe
Enter valid email

Password
••
Password must be at least 5 characters

Phone number
+ 123
Ex. 998901234567

Date of Birth *
05/02/2000

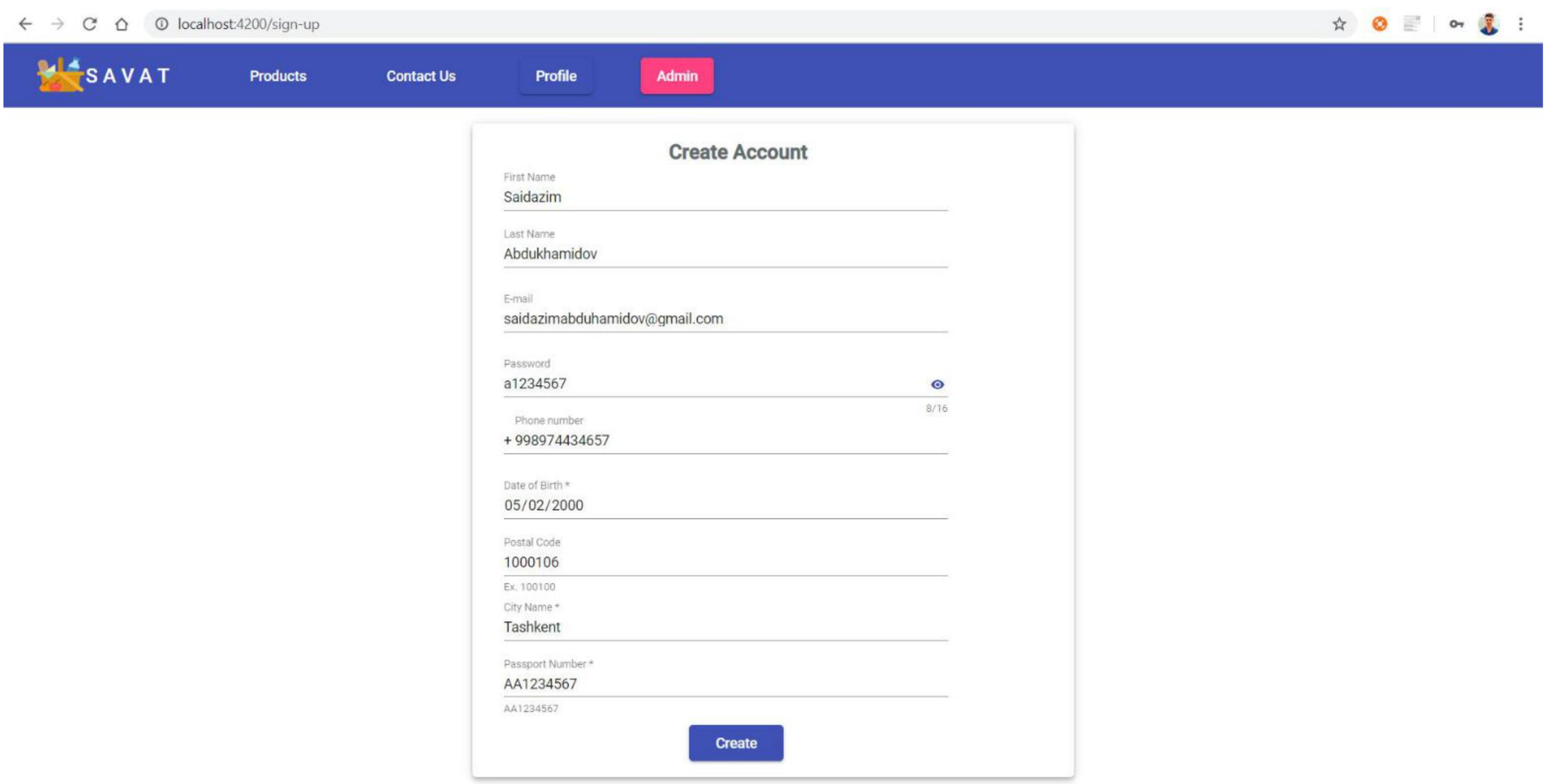
Postal Code
1
Postal code must be 7 digits

City Name *
Tashkent

Passport Number *
AA
Ex. AA1234567

Create

Email is our login for **Entering** and it should be **at least 5 characters as Password**, Other input forms also have validations. With **errors** Create button **does not** work.



Create Account

First Name
Saidazim

Last Name
Abdukhamidov

E-mail
saidazimabduhamidov@gmail.com

Password
a1234567
8/16

Phone number
+ 998974434657

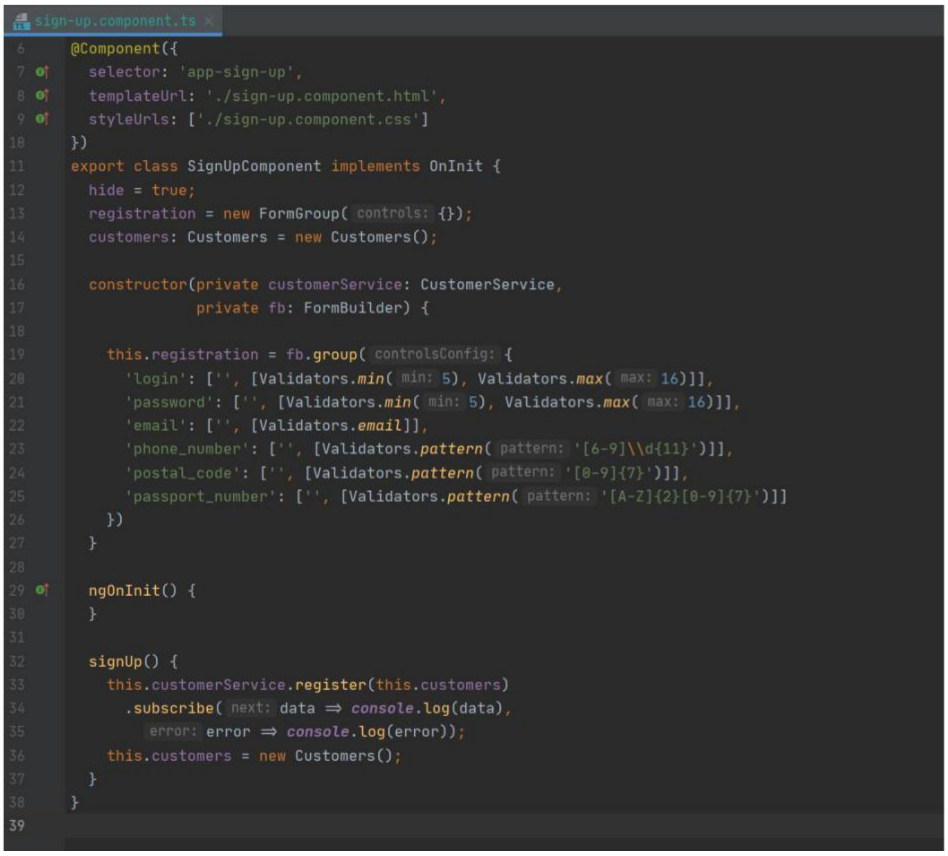
Date of Birth *
05/02/2000

Postal Code
1000106
Ex. 100100

City Name *
Tashkent

Passport Number *
AA1234567
AA1234567

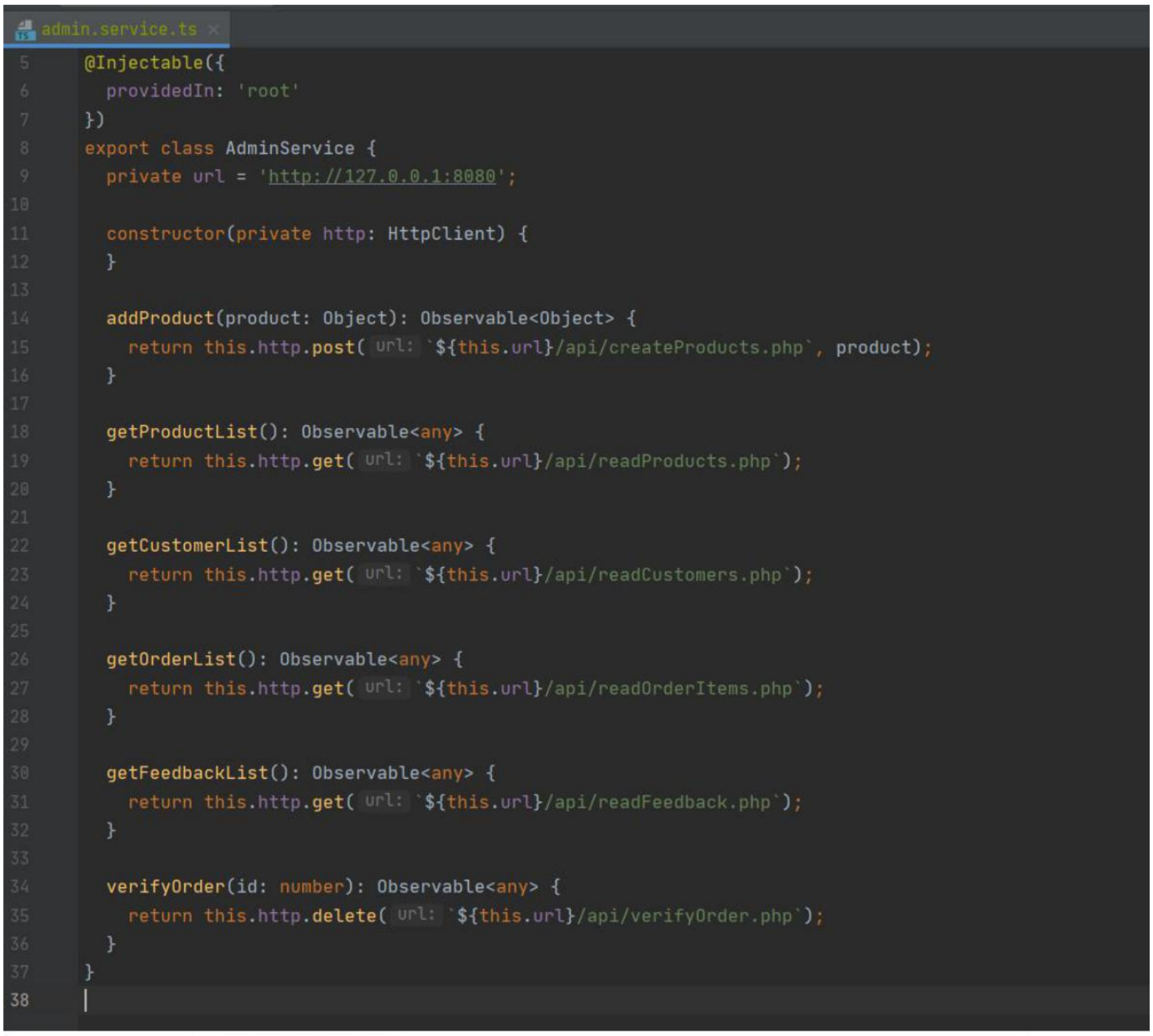
Create



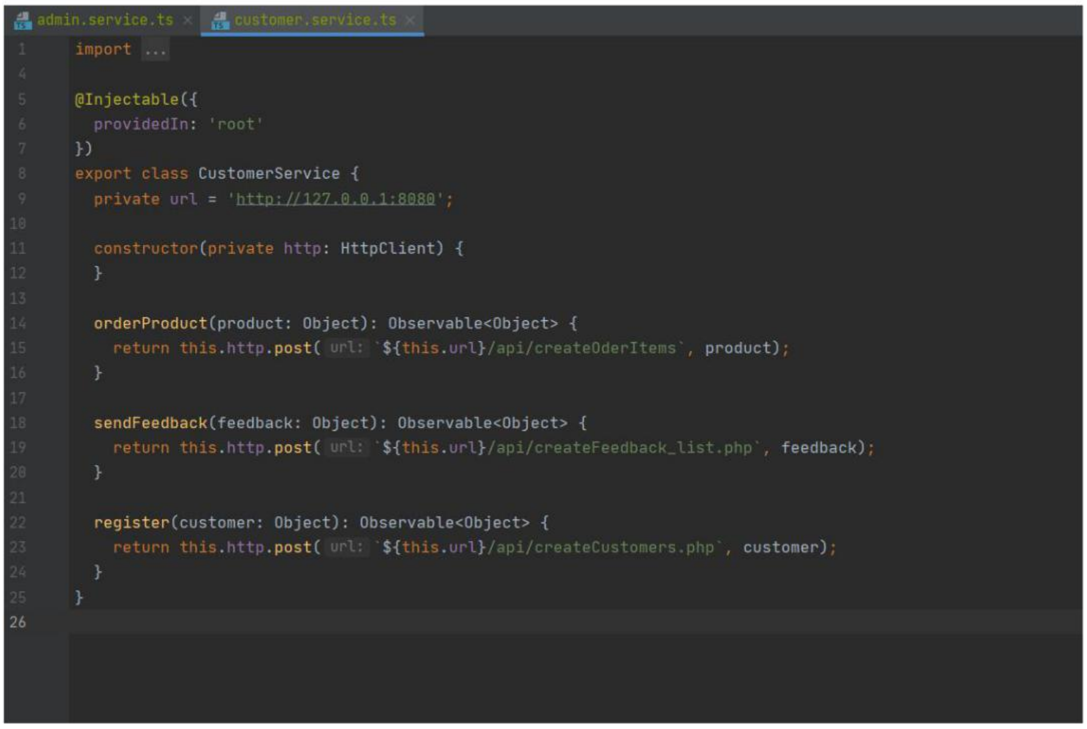
```
@Component({
  selector: 'app-sign-up',
  templateUrl: './sign-up.component.html',
  styleUrls: ['./sign-up.component.css']
})
export class SignUpComponent implements OnInit {
  hide = true;
  registration = new FormGroup( controls: {});
  customers: Customers = new Customers();

  constructor(private customerService: CustomerService,
              private fb: FormBuilder) {

    this.registration = fb.group( controlsConfig: {
      'login': ['', [Validators.min( min: 5), Validators.max( max: 16)]],
      'password': ['', [Validators.min( min: 5), Validators.max( max: 16)]],
      'email': ['', [Validators.email]],
      'phone_number': ['', [Validators.pattern( pattern: '[6-9]\\d{11}')]],
      'postal_code': ['', [Validators.pattern( pattern: '[0-9]{7}')]],
      'passport_number': ['', [Validators.pattern( pattern: '[A-Z]{2}[0-9]{7}')]]
    })
  }

  ngOnInit() {
  }

  signUp() {
    this.customerService.register(this.customers)
      .subscribe( next: data => console.log(data),
        error: error => console.log(error));
    this.customers = new Customers();
  }
}
```

There are our services:

admin.service.ts

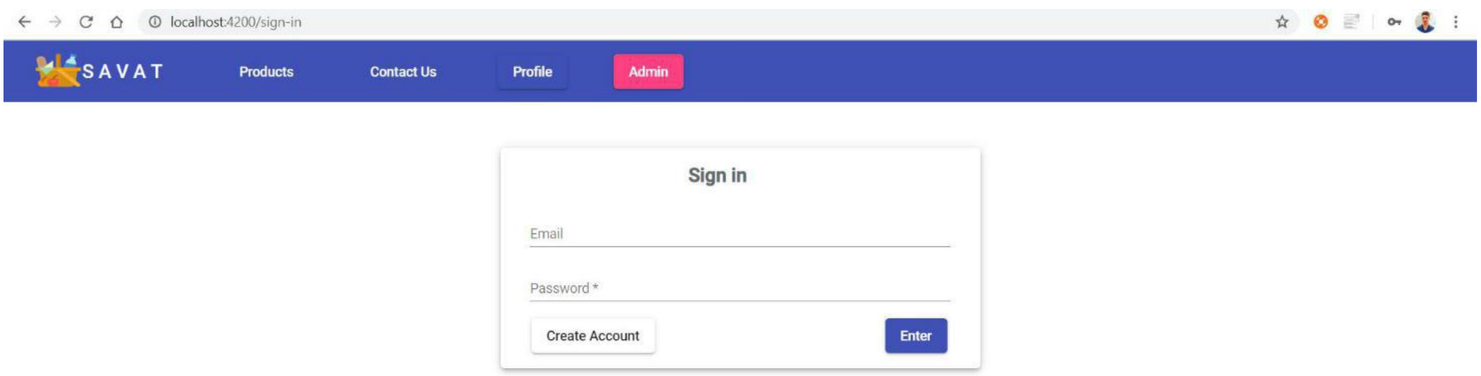


```
@Injectable({
  providedIn: 'root'
})
export class AdminService {
  private url = 'http://127.0.0.1:8080';

  constructor(private http: HttpClient) {
  }

  addProduct(product: Object): Observable<Object> {
    return this.http.post( url: `${this.url}/api/createProducts.php`, product);
  }

  getProductList(): Observable<any> {
    return this.http.get( url: `${this.url}/api/readProducts.php`);
  }

  getCustomerList(): Observable<any> {
    return this.http.get( url: `${this.url}/api/readCustomers.php`);
  }

  getOrderList(): Observable<any> {
    return this.http.get( url: `${this.url}/api/readOrderItems.php`);
  }

  getFeedbackList(): Observable<any> {
    return this.http.get( url: `${this.url}/api/readFeedback.php`);
  }

  verifyOrder(id: number): Observable<any> {
    return this.http.delete( url: `${this.url}/api/verifyOrder.php`);
  }
}

```



```
import ...

@Injectable({
  providedIn: 'root'
})
export class CustomerService {
  private url = 'http://127.0.0.1:8080';

  constructor(private http: HttpClient) {
  }

  orderProduct(product: Object): Observable<Object> {
    return this.http.post( url: `${this.url}/api/createOderItems`, product);
  }

  sendFeedback(feedback: Object): Observable<Object> {
    return this.http.post( url: `${this.url}/api/createFeedback_list.php`, feedback);
  }

  register(customer: Object): Observable<Object> {
    return this.http.post( url: `${this.url}/api/createCustomers.php`, customer);
  }
}

```

In the *savat/api* foler there are logic of our **PHP**, connecting to database, getting and setting data. For this reason next to our **URL** we add path of that files



Unfortunately, we could not make role-based authorization. For this reason, Login page does not work, but users can use this application without entering to own profile.