НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО

Факультет інформатики та обчислювальної техніки Кафедра інформатики та програмної інженерії

Звіт по лабораторній роботі №6

«Робота з проектом «SportShop» (продовження).

Створення функціонального модуля для адміністрування магазину.»

роботи з дисципліни: «Реактивне програмування»

Студент: Мєшков Андрій Ігорович

Група: ІП-15

Дата захисту роботи: _____

Викладач: доц. Полупан Юлія Вікторівна

Захищено з оцінкою: _____

Київ, 2024

ЗМІСТ

Опис модуля admin.module.ts з відповідною маршрутизацією для відображення компонентів з адміністрування 3

Специфікація JWT: основні поняття 6

Опис основних методів джерела даних, що використовуються при роботі адміністратора магазину «SportShop» 10

Опис основних методів репозиторія продуктів, що використовуються при роботі адміністратора магазину «SportShop» 13

Бібліотека Angular Material: основні поняття, призначення. Опис використовуваних в додатку функцій бібліотеки 16

Реалізація меню. Опис функцій таблиці товарів та таблиці замовлень магазину «SportShop» 22

ВИСНОВКИ 28

СПИСОК ДЖЕРЕЛ 30

Опис модуля admin.module.ts з відповідною маршрутизацією для відображення компонентів з адміністрування

Модуль AdminModule є основним модулем, який забезпечує функціонал адміністрування в додатку. Він відповідає за такі задачі, як аутентифікація користувачів, управління товарами та обробка замовлень. Цей модуль побудований з використанням **lazy loading**, що дозволяє завантажувати його тільки тоді, коли це необхідно, оптимізуючи продуктивність додатку. Для захисту доступу до адміністративної панелі використовується сервіс-охоронець AuthGuard, який гарантує, що лише авторизовані користувачі можуть отримати доступ до відповідних функцій.

Основними компонентами цього модуля є AuthComponent, AdminComponent, ProductTableComponent, ProductEditorComponent та OrderTableComponent. Компонент AuthComponent реалізує процес авторизації, дозволяючи адміністраторам увійти в систему. Після авторизації користувач потрапляє до AdminComponent, який є головним компонентом панелі адміністрування. У цій панелі доступні функції для перегляду та редагування списку товарів, а також для перегляду замовлень. Таблиця товарів реалізована в компоненті ProductTableComponent, а редактор товарів — у ProductEditorComponent, який дозволяє створювати нові товари або редагувати наявні. Для перегляду замовлень використовується компонент OrderTableComponent.

Маршрутизація в цьому модулі організована таким чином, що:

- Шлях /auth відкриває компонент авторизації AuthComponent і служить початковою точкою входу до модуля.

- Після успішної авторизації користувачі потрапляють на маршрут /main, який завантажує компонент AdminComponent. Цей маршрут захищений AuthGuard.
- У межах AdminComponent передбачені дочірні маршрути для управління товарами та замовленнями:
    - /products відкриває таблицю товарів (ProductTableComponent).
    - /products/:mode або /products/:mode/:id відкривають редактор товарів (ProductEditorComponent), де mode вказує на режим створення або редагування.
    - /orders відкриває таблицю замовлень (OrderTableComponent).
    - Будь-які невизначені шляхи перенаправляються до таблиці товарів.

У разі, якщо користувач переходить на невизначений маршрут в межах всього модуля, його автоматично перенаправляють на сторінку авторизації (/auth).

Додатково, у AdminModule імпортується MaterialFeatures для використання компонентів Angular Material, а також базові модулі Angular, такі як CommonModule, FormsModule, і маршрутизація через RouterModule.

Цей модуль інтегрується в основний додаток через модуль AppModule. У AppModule маршрут admin налаштований для завантаження AdminModule за допомогою **lazy loading**. Завдяки цьому адміністративна частина додатку завантажується лише тоді, коли користувач переходить до відповідного розділу. Це сприяє зменшенню обсягу даних, які потрібно завантажувати під час початкового завантаження додатку, що покращує загальну продуктивність системи.

src/app/admin/admin.module.ts

```
import { NgModule } from "@angular/core";
import { CommonModule } from "@angular/common";
import { FormsModule } from "@angular/forms";
import { RouterModule } from "@angular/router";
import { AuthComponent } from "./auth.component";
import { AdminComponent } from "./admin.component";
import { AuthGuard } from "./auth.guard";
import { MaterialFeatures } from "./material.module";
import { ProductTableComponent } from "./productTable.component";
import { ProductEditorComponent } from "./productEditor.component";
import { OrderTableComponent } from "./orderTable.component";
let routing = RouterModule.forChild([
    { path: "auth", component: AuthComponent },
    {
        path: "main", component: AdminComponent, canActivate: [AuthGuard],
        children: [
            { path: "products/:mode/:id", component: ProductEditorComponent },
            { path: "products/:mode", component: ProductEditorComponent },
            { path: "products", component: ProductTableComponent },
            { path: "orders", component: OrderTableComponent },
            { path: "**", redirectTo: "products" }
        ]
    },
    { path: "**", redirectTo: "auth" }
]);
@NgModule({
    imports: [CommonModule, FormsModule, routing, MaterialFeatures],
    declarations: [AuthComponent, AdminComponent, ProductTableComponent,
ProductEditorComponent, OrderTableComponent],
    providers: [AuthGuard]
})
export class AdminModule { }
```

Специфікація JWT: основні поняття

JWT (JSON Web Token) — це стандарт для створення токенів доступу, які використовуються для передачі даних між клієнтом і сервером в безпечний спосіб. Токени містять три частини:

1. **Header** (заголовок) — вказує тип токена (JWT) і алгоритм підпису (наприклад, HS256).
2. **Payload** (завантаження) — містить корисну інформацію (наприклад, ідентифікатор користувача).
3. **Signature** (підпис) — використовується для перевірки цілісності токена та його автентичності.

Аналіз коду authMiddleware.js

Цей файл реалізує middleware для Node.js, який додає механізм аутентифікації за допомогою JWT. Нижче наведено основні елементи і їх пояснення:

1. Константи

- APP_SECRET: секретний ключ, який використовується для підпису та перевірки токенів.
- USERNAME і PASSWORD: фіксовані облікові дані для входу.
- mappings: об'єкт, який визначає, які методи (get, post) і URL-адреси вимагають аутентифікації.

2. Функція requiresAuth

Ця функція перевіряє, чи потребує конкретний метод і URL-адреса авторизації. Вона використовує:

- mappings для визначення захищених маршрутів.

- Повертає true, якщо URL відповідає будь-якому захищеному маршруту, інакше false.

3. Middleware-логіка

- **Обробка /login:** Якщо клієнт надсилає запит на /login із методом POST, middleware перевіряє облікові дані. У разі успіху:
  - Генерується токен за допомогою jwt.sign().
  - У відповідь клієнту відправляється токен.
- **Обробка захищених маршрутів:**
  - Якщо URL вимагає авторизації, middleware перевіряє заголовок authorization.
  - Якщо токен є і він правильний, запит проходить далі.
  - У разі відсутності токена або помилкової перевірки сервер повертає код статусу 401 Unauthorized.

4. Генерація токена

Функція jwt.sign() використовується для створення токена. Аргументи:

- { data: USERNAME, expiresIn: "1h" }: корисне навантаження токена, де:
  - data: інформація, що передається.
  - expiresIn: час дії токена (1 година).
- APP_SECRET: секретний ключ для підпису.

5. Перевірка токена

Функція jwt.verify() перевіряє:

- Чи був токен підписаний коректно (APP_SECRET).
- Чи не закінчився термін дії токена.

Поток даних у додатку

1. **Аутентифікація:**
    - Клієнт надсилає облікові дані на /login.
    - У разі успіху отримує токен.
2. **Захищені запити:**
    - Клієнт додає токен у заголовок Authorization при виконанні запитів.
    - Сервер перевіряє токен і надає доступ до захищених ресурсів.
3. **Неавторизовані запити:**
    - Якщо токен відсутній або недійсний, сервер повертає код 401.

Особливості реалізації

- **Безпека:** Використання секретного ключа гарантує, що тільки сервер може підписувати і перевіряти токени.
- **Масштабованість:** Токени не потребують збереження на сервері, оскільки вони самодостатні.
- **Гнучкість:** Middleware дозволяє легко додавати нові захищені маршрути в mappings.

authMiddleware.js

```
const jwt = require("jsonwebtoken");
const APP_SECRET = "myappsecret";
const USERNAME = "admin";
const PASSWORD = "secret";
const mappings = {
    get: ["/api/orders", "/orders"],
    post: ["/api/products", "/products", "/api/categories", "/categories"]
}
function requiresAuth(method, url) {
    return (mappings[method.toLowerCase()] || [])
    .find(p => url.startsWith(p)) !== undefined;
}
module.exports = function (req, res, next) {
    if (req.url.endsWith("/login") && req.method == "POST") {
```

```
        if (req.body && req.body.name == USERNAME &&
            req.body.password==PASSWORD) {
            let token = jwt.sign({ data: USERNAME, expiresIn: "1h" }, APP_SECRET);
            res.json({ success: true, token: token });
        } else {
            res.json({ success: false });
        }
        res.end();
        return;
    } else if (requiresAuth(req.method, req.url)) {
        let token = req.headers["authorization"] || "";
        if (token.startsWith("Bearer<")) {
            token = token.substring(7, token.length - 1);
            try {
                jwt.verify(token, APP_SECRET);
                next();
                return;
            } catch (err) { }
        }
        res.statusCode = 401;
        res.end();
        return;
    }
    next();
}
```

Опис основних методів джерела даних, що використовуються при роботі адміністратора магазину «SportShop»

Клас RestDataSource є основним джерелом даних для роботи з сервером в адміністративному інтерфейсі магазину «SportShop». Він реалізований як Angular-сервіс, що надає методи для виконання HTTP-запитів до API. Основні функції включають отримання, створення, оновлення та видалення продуктів і замовлень, а також аутентифікацію адміністратора.

Опис методів джерела даних RestDataSource

Сервіс RestDataSource забезпечує взаємодію адміністратора магазину «SportShop» із REST API. Основні методи:

1. **Аутентифікація:**
   - **authenticate**(user: string, pass: string): Надсилає POST-запит на /login, отримує токен при успішному вході.
2. **Операції з продуктами:**
   - **getProducts**(): Надсилає GET-запит на /products, повертає список продуктів.
   - **saveProduct**(product: Product): Зберігає новий продукт через POST-запит.
   - **updateProduct**(product: Product): Оновлює продукт за допомогою PUT-запиту.
   - **deleteProduct**(id: number): Видаляє продукт через DELETE-запит.
3. **Операції із замовленнями:**
   - **getOrders**(): Отримує список замовлень через GET-запит.
   - **saveOrder**(order: Order): Надсилає POST-запит на /orders, зберігаючи нове замовлення.

- **updateOrder**(order: Order): Оновлює замовлення через PUT-запит.
- **deleteOrder**(id: number): Видаляє замовлення через DELETE-запит.

4. **Авторизація:**
   - **getOptions**(): Додає токен до заголовків запитів для захищених операцій.

Сервіс автоматизує обробку даних і забезпечує захищений доступ до ресурсів.

src/app/model/rest.datasource.ts

```
import { Injectable } from "@angular/core";
import { HttpClient } from "@angular/common/http";
import { map, Observable } from "rxjs";
import { Product } from "./product.model";
import { Order } from "./order.model";
import { HttpHeaders } from '@angular/common/http';
const PROTOCOL = "http";
const PORT = 3500;
@Injectable()
export class RestDataSource {
    baseUrl: string;
    auth_token?: string;
    constructor(private http: HttpClient) {
        this.baseUrl = `${PROTOCOL}://${location.hostname}:${PORT}/`;
    }
    getProducts(): Observable<Product[]> {
        return this.http.get<Product[]>(this.baseUrl + "products");
    }
    saveOrder(order: Order): Observable<Order> {
        return this.http.post<Order>(this.baseUrl + "orders", order);
    }
    authenticate(user: string, pass: string): Observable<boolean> {
        return this.http.post<any>(this.baseUrl + "login", {
            name: user, password: pass
        }).pipe(map(response => {
            this.auth_token = response.success ? response.token : null;
            return response.success;
        }));
    }
    saveProduct(product: Product): Observable<Product> {
        return this.http.post<Product>(this.baseUrl + "products", product,
this.getOptions());
```

```
    }
    updateProduct(product: Product): Observable<Product> {
        return this.http.put<Product>(`${this.baseUrl}products/${product.id}`,
product, this.getOptions());
    }
    deleteProduct(id: number): Observable<Product> {
        return this.http.delete<Product>(`${this.baseUrl}products/${id}`,
this.getOptions());
    }
    getOrders(): Observable<Order[]> {
        return this.http.get<Order[]>(this.baseUrl + "orders", this.getOptions());
    }
    deleteOrder(id: number): Observable<Order> {
        return this.http.delete<Order>(`${this.baseUrl}orders/${id}`,
this.getOptions());
    }
    updateOrder(order: Order): Observable<Order> {
        return this.http.put<Order>(`${this.baseUrl}orders/${order.id}`, order,
this.getOptions());
    }
    private getOptions() {
        return {
            headers: new HttpHeaders({
                "Authorization": `Bearer<${this.auth_token}>`
            })
        }
    }
}
```

Опис основних методів репозиторія продуктів, що використовуються при роботі адміністратора магазину «SportShop»

Додаток SportShop включає низку Angular-компонентів, кожен із яких відповідає за окремий функціонал магазину. Ось перелік основних компонентів із описом їхнього призначення та використання.

Репозиторій ProductRepository є проміжним шаром між джерелом даних (RestDataSource) і додатком. Він обробляє продукти та категорії, забезпечуючи зручний інтерфейс для роботи адміністратора магазину.

Основні методи:

1. **Конструктор:**
    - Ініціалізує продукти та категорії шляхом підписки на метод **getProducts** джерела даних.
    - Категорії формуються на основі отриманих продуктів, видаляючи повторення і сортування.
2. **Отримання продуктів:**
    - **getProducts**(category?: string): Product[]: Повертає список продуктів. Якщо передано категорію, фільтрує продукти за нею.
3. **Отримання продукту за ідентифікатором:**
    - **getProduct**(id: number): Product | undefined: Знаходить продукт за його id. Якщо такого продукту немає, повертає undefined.
4. **Отримання категорій:**
    - **getCategories**(): string[]: Повертає список доступних категорій продуктів.
5. **Збереження продукту:**

- **saveProduct**(product: Product):
  - Якщо id продукту не вказано або дорівнює 0, додає новий продукт через метод saveProduct джерела даних.
  - Інакше оновлює існуючий продукт за допомогою updateProduct і замінює його в локальному списку.

6. **Видалення продукту:**
   - **deleteProduct**(id: number): Видаляє продукт за допомогою методу deleteProduct джерела даних. Успішне видалення також видаляє продукт із локального списку.

Призначення репозиторію:

- **Буферизація даних:** Репозиторій зберігає локальну копію продуктів та категорій, мінімізуючи кількість запитів до сервера.
- **Інкапсуляція:** Спрощує доступ до джерела даних, абстрагуючи логіку запитів.
- **Зручний інтерфейс:** Надає методи для отримання продуктів, категорій, а також для збереження та видалення продуктів.

Цей репозиторій є ключовим елементом для організації ефективної взаємодії адміністратора з даними в магазині «SportShop».

src/app/model/product.repository.ts

```
import { Injectable } from "@angular/core";
import { Product } from "./product.model";
import { RestDataSource } from "./rest.datasource";
@Injectable()
export class ProductRepository {
    private products: Product[] = [];
    private categories: string[] = [];
    constructor(private dataSource: RestDataSource) {
        dataSource.getProducts().subscribe(data => {
            this.products = data;
            this.categories = data.map(p => p.category ?? "(None)")
            .filter((c, index, array) => array.indexOf(c) == index).sort();
```

```
        });
    }
    getProducts(category?: string): Product[] {
        return this.products
        .filter(p => category == undefined || category == p.category);
    }
    getProduct(id: number): Product | undefined {
        return this.products.find(p => p.id == id);
    }
    getCategories(): string[] {
        return this.categories;
    }
    saveProduct(product: Product) {
        if (product.id == null || product.id == 0) {
            this.dataSource.saveProduct(product)
                .subscribe(p => this.products.push(p));
        } else {
            this.dataSource.updateProduct(product)
                .subscribe(p => {
                    this.products.splice(this.products.findIndex(p => p.id ==
product.id), 1, product);
                });
        }
    }
    deleteProduct(id: number) {
        this.dataSource.deleteProduct(id).subscribe(p => {
            this.products.splice(this.products.findIndex(p => p.id == id), 1);
        })
    }
}
```

Бібліотека Angular Material: основні поняття, призначення. Опис використовуваних в додатку функцій бібліотеки

Angular Material — це набір компонентів користувацького інтерфейсу, які відповідають вимогам специфікацій Material Design, розроблених компанією Google. Angular Material забезпечує готові до використання UI компоненти, які дозволяють розробникам швидко створювати стильні, адаптивні та функціональні інтерфейси.

Основне призначення Angular Material:

1. **Покращення користувацького досвіду:** Завдяки готовим стилям та інтерактивним елементам, розробники можуть створювати сучасні інтерфейси з мінімальними зусиллями.
2. **Мобільність та адаптивність:** Компоненти автоматично адаптуються під різні розміри екрану, що важливо для підтримки мобільних версій веб-додатків.
3. **Простота інтеграції:** Компоненти Angular Material безшовно інтегруються з фреймворком Angular, маючи підтримку типів та шаблонів, що значно зменшує обсяг ручної роботи для розробників.

Використовувані в додатку функції бібліотеки Angular Material:

MatToolbarModule (Тулбар): Використовуються для створення навігаційної панелі вгорі сторінки. У додатку цей компонент забезпечує заголовок "SportShop Administration" в admin.component, а також кнопку для відкриття/закриття бокового меню (sidenav).

MatSidenavModule (Бокова панель): Цей компонент використовується для створення бокового меню, яке можна приховувати та показувати за

допомогою кнопки. У додатку він містить кнопки для навігації між різними розділами адміністративної панелі (наприклад, "Products", "Orders").

MatButtonModule (Кнопки): Використовуються для створення кнопок у інтерфейсі. У додатку ці кнопки використовуються для навігації та взаємодії з елементами інтерфейсу, такими як створення нових продуктів або виконання дій з замовленнями.

MatTableModule (Таблиці): Забезпечує створення таблиць з підтримкою таких функцій, як пагінація, сортування та фільтрація. У додатку цей компонент використовується для відображення списків продуктів та замовлень. Наприклад, у productTable відображаються продукти з можливістю видалення або редагування.

MatPaginatorModule (Пагінація): Цей компонент додає пагінацію до таблиць для зручнішого перегляду великих обсягів даних. У додатку пагінація використовується в таблиці продуктів для розбиття даних на сторінки.

MatFormFieldModule (Форми): Використовується для стилізації форм і полів вводу. У додатку поля введення для створення або редагування продуктів використовують цей компонент, що забезпечує естетичне відображення полів вводу.

MatInputModule (Введення даних): Додає підтримку стилізованих полів вводу, таких як текстові поля, числові значення тощо. У додатку поля вводу для продуктів, таких як "Name", "Category", "Price" використовують matInput.

MatCheckboxModule (Перевірочні поля): Використовуються для створення перевірочних полів, наприклад, для фільтрації замовлень за станом доставки. У додатку цей компонент дозволяє адміністраторам відфільтровувати відображення доставлених замовлень.

Angular Material значно полегшує розробку сучасних та інтерактивних інтерфейсів, надаючи компоненти, що відповідають специфікаціям Material Design. У додатку SportShop використовуються компоненти для створення

зручних форм, таблиць, навігації та взаємодії з користувачем, що забезпечує гарний вигляд і зручність в користуванні адміністратору.

src/app/admin/material.module.ts

```
import { NgModule } from "@angular/core";
import { MatToolbarModule } from "@angular/material/toolbar";
import { MatSidenavModule } from "@angular/material/sidenav";
import { MatIconModule } from '@angular/material/icon';
import { MatDividerModule } from '@angular/material/divider';
import { MatButtonModule } from "@angular/material/button";
import { MatTableModule } from "@angular/material/table";
import { MatPaginatorModule } from "@angular/material/paginator";
import { MatFormFieldModule } from '@angular/material/form-field';
import { MatInputModule } from '@angular/material/input';
import { MatCheckboxModule } from '@angular/material/checkbox';
const features: any[] = [MatToolbarModule,MatSidenavModule,MatIconModule,
MatDividerModule, MatButtonModule, MatTableModule,
MatPaginatorModule,MatFormFieldModule, MatInputModule, MatCheckboxModule];
@NgModule({
    imports: [features],
    exports: [features]
})
export class MaterialFeatures {}
```

src/app/admin/admin.component.html

```
<mat-toolbar color="primary">
    <div>
        <button mat-icon-button *ngIf="sidenav.mode === 'over'"
(click)="sidenav.toggle()">
            <mat-icon *ngIf="!sidenav.opened">menu</mat-icon>
            <mat-icon *ngIf="sidenav.opened">close</mat-icon>
        </button>
        SportShop Administration
    </div>
</mat-toolbar>
<mat-sidenav-container>
    <mat-sidenav #sidenav="matSidenav" class="mat-elevation-z8">
        <div>
            <button mat-button class="menu-button"
                routerLink="/admin/main/products"
                routerLinkActive="mat-accent"
                (click)="sidenav.close()">
                <mat-icon>shopping_cart</mat-icon>
                <span>Products</span>
            </button>
        </div>
        <div>
            <button mat-button class="menu-button"
```

```
                routerLink="/admin/main/orders"
                routerLinkActive="mat-accent"
                (click)="sidenav.close()">
                <mat-icon>local_shipping</mat-icon>
                <span>Orders</span>
            </button>
        </div>
        <mat-divider></mat-divider>
        <button mat-button class="menu-button logout" (click)="logout()">
            <mat-icon>logout</mat-icon>
            <span>Logout</span>
        </button>
    </mat-sidenav>
    <mat-sidenav-content>
        <div class="content">
            <router-outlet></router-outlet>
        </div>
    </mat-sidenav-content>
</mat-sidenav-container>
```

src/app/admin/productTable.component.html

```
<table mat-table [dataSource]="dataSource">
    <mat-text-column name="id"></mat-text-column>
    <mat-text-column name="name"></mat-text-column>
    <mat-text-column name="category"></mat-text-column>
    <ng-container matColumnDef="price">
        <th mat-header-cell *matHeaderCellDef>Price</th>
        <td mat-cell *matCellDef="let item"> {{item.price | currency:"USD"}} </td>
    </ng-container>
    <ng-container matColumnDef="buttons">
        <th mat-header-cell *matHeaderCellDef></th>
        <td mat-cell *matCellDef="let p">
    <button mat-flat-button color="accent" (click)="deleteProduct(p.id)">
        Delete
    </button>
    <button mat-flat-button color="warn"
[routerLink]="['/admin/main/products/edit', p.id]">
        Edit
    </button>
    </ng-container>
    <tr mat-header-row *matHeaderRowDef="colsAndRows"></tr>
    <tr mat-row *matRowDef="let row; columns: colsAndRows"></tr>
</table>
<div class="bottom-box">
    <button mat-flat-button color="primary"
routerLink="/admin/main/products/create">
        Create New Product
    </button>
    <mat-paginator [pageSize]="5" [pageSizeOptions]="[3, 5, 10]">
    </mat-paginator>
```

```
</div>
```

src/app/admin/productEditor.component.html

```
<h3 class="heading">{{editing ? "Edit" : "Create"}} Product</h3>
<form (ngSubmit)="save()">
    <mat-form-field *ngIf="editing">
        <span>ID</span>
        <input class="form-control" name="id" [(ngModel)]="product.id" disabled />
    </mat-form-field>
    <mat-form-field>
        <span>Name</span>
        <input class="form-control" name="name" [(ngModel)]="product.name" />
    </mat-form-field>
    <mat-form-field>
        <span>Category</span>
        <input class="form-control" name="category" [(ngModel)]="product.category"
/>
    </mat-form-field>
    <mat-form-field>
        <span>Description</span>
        <input class="form-control" name="description"
[(ngModel)]="product.description"/>
    </mat-form-field>
    <mat-form-field>
        <span>Price</span>
        <input class="form-control" name="price" [(ngModel)]="product.price" />
    </mat-form-field>
    <div>
        <button type="submit" mat-flat-button color="primary">
            {{editing ? "Save" : "Create"}}
        </button>
        <button type="reset" mat-stroked-button  routerLink="/admin/main/products">
            Cancel
        </button>
    </div>
</form>
```

src/app/admin/orderTable.component.html

```

<mat-checkbox [(ngModel)]="includeShipped">Display Shipped Orders</mat-checkbox>
<div class="filter-container">
    <button *ngFor="let letter of 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'.split('')"
            (click)="changeLetter(letter)">
        {{ letter }}
    </button>
    <button (click)="changeLetter('')">All</button>
</div>
<table class="orders" mat-table [dataSource]="dataSource">
```

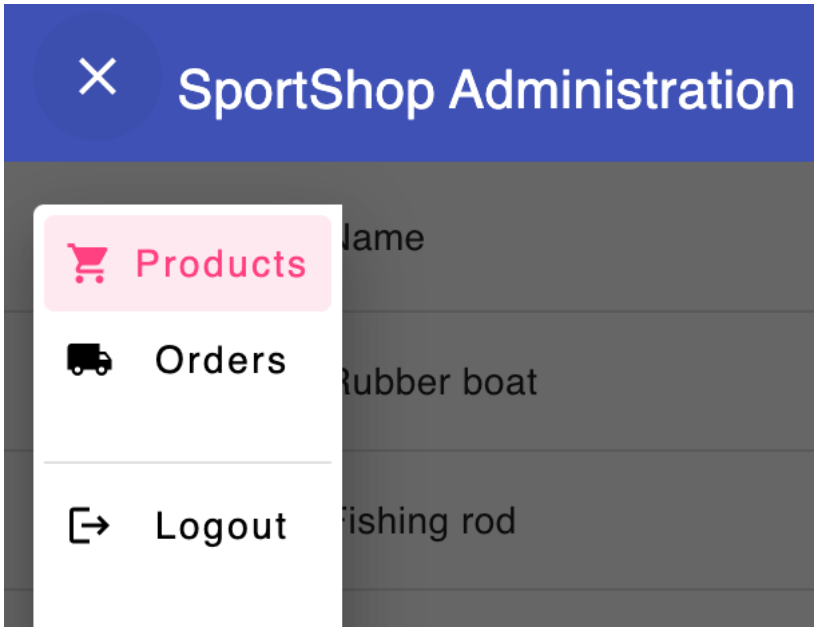
```
    <mat-text-column name="name"></mat-text-column>
    <mat-text-column name="zip"></mat-text-column>
    <ng-container matColumnDef="cart_p">
        <th mat-header-cell *matHeaderCellDef></th>
        <td mat-cell *matCellDef="let order">
            <table mat-table [dataSource]="order.cart.lines">
                <ng-container matColumnDef="p">
                    <th mat-header-cell *matHeaderCellDef>Product</th>
                    <td mat-cell *matCellDef="let line">{{ line.product.name
}}</td>
                </ng-container>
                <tr mat-header-row *matHeaderRowDef="['p']"></tr>
                <tr mat-row *matRowDef="let row; columns: ['p']"></tr>
            </table>
        </td>
    </ng-container>
    <ng-container matColumnDef="cart_q">
        <th mat-header-cell *matHeaderCellDef></th>
        <td mat-cell *matCellDef="let order">
            <table mat-table [dataSource]="order.cart.lines">
                <ng-container matColumnDef="q">
                    <th mat-header-cell *matHeaderCellDef>Quantity</th>
                    <td mat-cell *matCellDef="let line">{{ line.quantity }}</td>
                </ng-container>
                <tr mat-header-row *matHeaderRowDef="['q']"></tr>
                <tr mat-row *matRowDef="let row; columns: ['q']"></tr>
            </table>
        </td>
    </ng-container>
    <ng-container matColumnDef="buttons">
        <th mat-header-cell *matHeaderCellDef>Actions</th>
        <td mat-cell *matCellDef="let o">
        <button mat-flat-button color="primary" (click)="toggleShipped(o)">
            {{ o.shipped ? "Unship" : "Ship" }}
        </button>
        <button mat-flat-button color="warn" (click)="delete(o.id)">
            Delete
        </button>
    </ng-container>
    <tr mat-header-row *matHeaderRowDef="colsAndRows"></tr>
    <tr mat-row *matRowDef="let row; columns: colsAndRows"></tr>
    <tr class="mat-row" *matNoDataRow>
        <td class="mat-cell no-data" colspan="4">No orders to display</td>
    </tr>
</table>
```

Реалізація меню. Опис функцій таблиці товарів та таблиці замовлень магазину «SportShop»

В додатку використовується директива **CounterDirective**, яка забезпечує динамічне рендеринг шаблонів на основі числового значення.

1. Реалізація меню адміністратора:

Меню адміністратора складається з бокового меню (sidenav), яке містить кнопки для навігації між різними розділами, такими як «Продукти» та «Замовлення», а також кнопку для виходу з системи. Кожен розділ має свій маршрут, за яким користувач може перейти для перегляду відповідної інформації. Кнопка «Logout» дозволяє вийти з системи, очищаючи дані автентифікації та перенаправляючи користувача на головну сторінку.



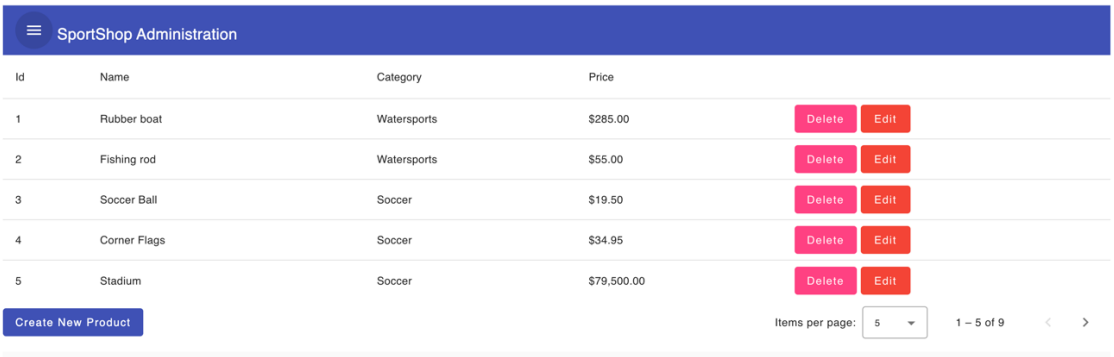
src/app/admin/admin.component.ts

```
import { Component } from "@angular/core";
import { Router } from "@angular/router";
import { AuthService } from "../model/auth.service";
@Component({
    templateUrl: "./admin.component.html"
```

```
})
export class AdminComponent {
    constructor(private auth: AuthService, private router: Router) { }
    logout() {
        this.auth.clear();
        this.router.navigateByUrl("/");
    }
}
```

2. Функціональність таблиці товарів:

Таблиця товарів надає користувачу можливість переглядати всі доступні продукти. Вона включає кілька колонок, таких як ID, назва, категорія, ціна та дії (редагування та видалення товару). Дані для таблиці надходять з репозиторію продуктів і відображаються через компонент MatTableDataSource, що дозволяє динамічно змінювати відображення даних. Для покращення користувацького досвіду, таблиця має пагінацію, яка дозволяє розділити список товарів на сторінки. Окрім цього, кожен продукт можна видалити, що викликає відповідну функцію у репозиторії продуктів.



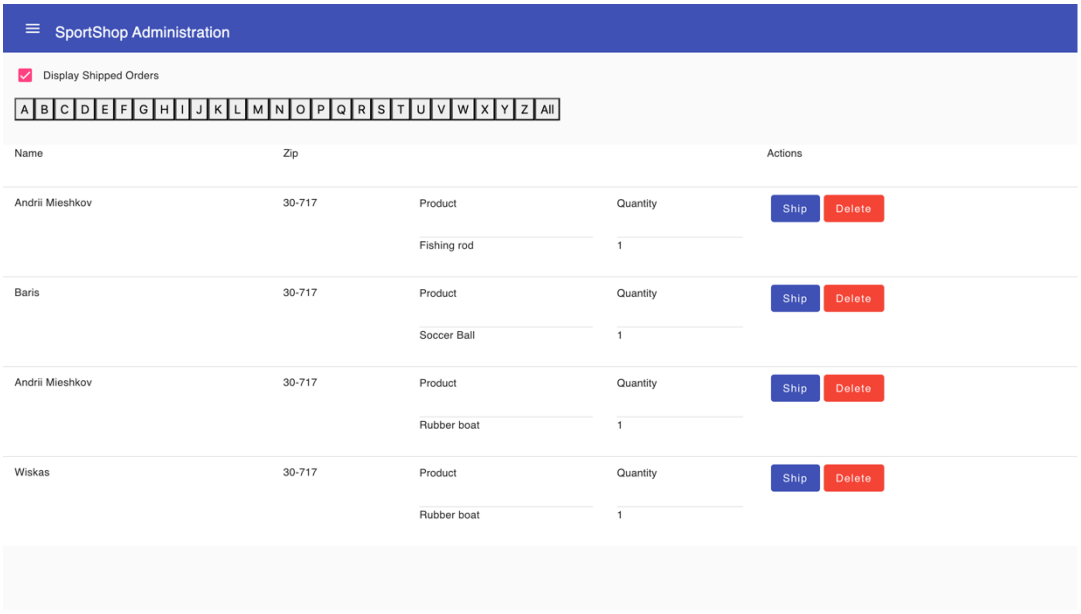
src/app/admin/productTable.component.ts

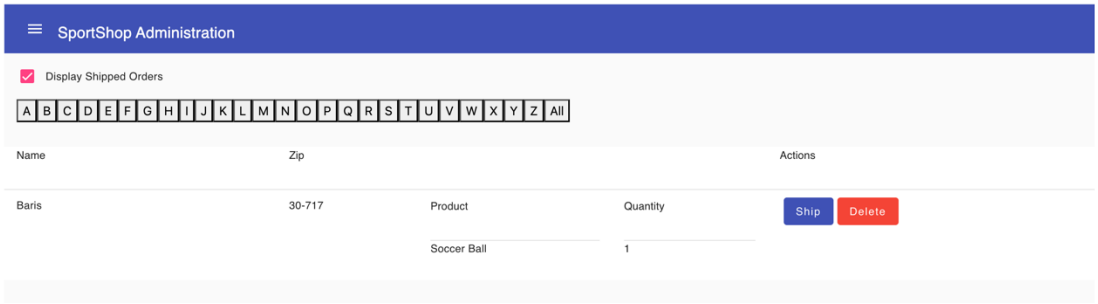
```
import { Component, IterableDiffer, IterableDiffers, ViewChild } from
"@angular/core"
import { MatTableDataSource } from "@angular/material/table";
import { Product } from "../model/product.model";
import { ProductRepository } from "../model/product.repository";
import { MatPaginator } from "@angular/material/paginator";
@Component({
    templateUrl: "./productTable.component.html"
})
export class ProductTableComponent {
```

```
    colsAndRows: string[] = ['id', 'name', 'category', 'price', 'buttons'];
    dataSource: MatTableDataSource<Product>;
    differ: IterableDiffer<Product>;

    @ViewChild(MatPaginator)
    paginator? : MatPaginator

    constructor(private repository: ProductRepository, differs: IterableDiffers) {
        this.dataSource = new
MatTableDataSource<Product>(this.repository.getProducts());
        this.differ = differs.find(this.repository.getProducts()).create();
    }
    ngDoCheck() {
        let changes = this.differ?.diff(this.repository.getProducts());
        if (changes != null) {
            this.dataSource.data = this.repository.getProducts();
        }
    }
    ngAfterViewInit() {
        if (this.paginator) {
            this.dataSource.paginator = this.paginator;
        }
    }
    deleteProduct(id: number) {
        this.repository.deleteProduct(id);
    }
}
```

3. Функціональність таблиці замовлень:

Таблиця замовлень дозволяє адміністраторам переглядати список замовлень, що були зроблені покупцями. Таблиця містить інформацію про замовлення, включаючи ім'я покупця, поштовий індекс, продукти в кошику, кількість одиниць кожного товару та кнопки для виконання дій (наприклад, зміна статусу доставки або видалення замовлення). Окрім того, користувач може фільтрувати замовлення за першою літерою імені покупця та за статусом доставки (показувати або приховувати відвантажені замовлення). Також є функція зміни статусу доставки для кожного замовлення, що дозволяє відмічати замовлення як відправлені або скасовувати цей статус.





src/app/admin/orderTable.component.ts

```
import { Component, IterableDiffer, IterableDiffers } from "@angular/core";
import { MatTableDataSource } from "@angular/material/table";
import { Order } from "../model/order.model";
import { OrderRepository } from "../model/order.repository";
@Component({
    templateUrl: "./orderTable.component.html"
})
export class OrderTableComponent {
    colsAndRows: string[] = ['name', 'zip','cart_p','cart_q', 'buttons'];
    dataSource: MatTableDataSource<Order>;
    differ: IterableDiffer<Order>;
    selectedLetter: string = '';
    constructor(private repository: OrderRepository, differs: IterableDiffers) {
        this.dataSource = new
MatTableDataSource<Order>(this.repository.getOrders(this.selectedLetter));
        this.differ =
differs.find(this.repository.getOrders(this.selectedLetter)).create();
        this.dataSource.filter = "true";
        this.dataSource.filterPredicate = (order, include) => {
            return !order.shipped || include.toString() == "true"
```

```
        };
    }
    get includeShipped(): boolean {
        return this.dataSource.filter == "true";
    }
    set includeShipped(include: boolean) {
        this.dataSource.filter = include.toString()
    }
    changeLetter(newLetter: string) {
        this.selectedLetter = newLetter;
        this.updateTableData();
    }
    toggleShipped(order: Order) {
        order.shipped = !order.shipped;
        this.repository.updateOrder(order);
    }
    delete(id: number) {
        this.repository.deleteOrder(id);
    }
    ngDoCheck() {
        let changes = this.differ?.diff(this.repository.getOrders());
        if (changes != null) {
            this.dataSource.data = this.repository.getOrders();
        }
    }

    private updateTableData() {
        this.dataSource.data = this.repository.getOrders(this.selectedLetter);
    }
}
```

4. Загальна структура:

Обидві таблиці використовують MatTableDataSource для управління даними таблиці та MatPaginator для пагінації. Вони дозволяють адміністраторам ефективно переглядати великий обсяг інформації, сортувати та фільтрувати дані, а також здійснювати необхідні дії з кожним елементом (наприклад, видаляти або редагувати продукти, змінювати статус замовлень). Використання MatCheckbox для фільтрації замовлень за станом доставки дозволяє швидко отримати потрібні дані.

Таким чином, реалізація таблиць товарів і замовлень вимагає використання компонентів Angular Material для побудови інтерфейсу, а також динамічної роботи з даними, що зберігаються в репозиторіях.

ВИСНОВКИ

У ході виконання лабораторної роботи було реалізовано функціональний модуль для адміністрування магазину «SportShop», що включає основні елементи адміністрування товарів та замовлень. Важливими етапами виконання роботи стали:

1. **Розробка модуля адміністрування (admin.module.ts)**: Було організовано маршрутизацію для відображення відповідних компонентів адміністрування, таких як управління товарами та замовленнями, що дозволяє зручно керувати даними магазину через інтерфейс адміністратора.
2. **Використання JWT для аутентифікації**: Для захисту доступу до адміністративних функцій був реалізований механізм JSON Web Token (JWT), який дозволяє користувачам авторизуватись у системі та отримувати доступ лише до дозволених ресурсів, забезпечуючи таким чином безпеку застосунку.
3. **Реалізація джерел даних та репозиторіїв**: Було створено методи для роботи з джерелами даних та репозиторіями продуктів і замовлень, що дає можливість ефективно зберігати, оновлювати та видаляти товари та замовлення, а також забезпечувати фільтрацію та пагінацію при їх перегляді.
4. **Використання бібліотеки Angular Material**: Для реалізації інтерфейсу були використані компоненти бібліотеки Angular Material, зокрема для побудови таблиць товарів і замовлень, а також для створення зручного меню адміністратора. Компоненти, такі як mat-toolbar, mat-sidenav, mat-table, mat-paginator, забезпечили сучасний та функціональний інтерфейс.

5. **Реалізація таблиць для товарів та замовлень**: Було розроблено таблиці для відображення даних товарів і замовлень з можливістю фільтрації, сортування та пагінації. Це дозволяє адміністраторам ефективно управляти великими обсягами інформації, швидко знаходити необхідні записи та виконувати операції редагування чи видалення.

Загалом, виконання лабораторної роботи дозволило набуття практичних навичок у розробці веб-додатків з використанням Angular, JWT для безпеки, а також компонентів бібліотеки Angular Material для створення сучасного інтерфейсу. Це сприяє розвитку знань у галузі розробки реактивних додатків і забезпечення їх функціональності та безпеки.

СПИСОК ДЖЕРЕЛ

1. Документація Angular. URL: https://v17.angular.io/docs
2. Документація Nodejs. URL: https://nodejs.org/docs/latest/api/
3. Документація Firebase. URL: https://firebase.google.com/docs?hl=en