**ДОДАТОК**

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ

СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт по лабораторній роботі No 4

Створення проекту «Blog».

З дисципліни: «Реактивне програмування»

Студент: Трофимов Данило Олегович

Група: ІП-02

Дата захисту роботи:

Викладач: доц. Полупан Юлія Вікторівна

Захищено з оцінкою:

Київ, 2023

# **ЗМІСТ**

[**ЗМІСТ** 2](#_Toc148684176)

[**Загальний хід роботи:** 3](#_Toc148684178)

[**1)Модель даних у проєкті «Todo»** 3](#_Toc148684179)

[**2)Пакет Angular Material: Призначення та використання.** 4](#_Toc148684180)

[**3)Двосторонні прив’язки даних: призначення, використання** 6](#_Toc148684181)

[**4)Функція filter у JavaScript: призначення, застосування, приклади** 7](#_Toc148684182)

[**5)** **Посилання на розгорнутий додаток** 8](#_Toc148684183)

[**Висновки** 9](#_Toc148684184)

[**Список літератури** 10](#_Toc148684185)

# **Загальний хід роботи:**

1. Згідно з інструкціями ініціалізовано angular проєкт з назвою Blog
2. Було опрацьовано завдання по створенню проєкту Blog
3. Було розгорнуто дотаток на платформі Firebase

# **Детальний огляд компоненту post**

Спочатку проведемо дослідження файлу post.component.ts:

Цей код представляє собою шаблон компонента PostComponent в Angular, і він визначає, як дані будуть відображатися на сторінці при використанні цього компонента. Розглянемо кожен елемент в шаблоні:

* import { Component, Input, OnInit, OnDestroy, Output, EventEmitter } from '@angular/core'; Ця рядок імпортує необхідні класи та функції з модуля @angular/core. Зокрема, Component використовується для створення Angular компоненту, Input та Output для роботи з вхідними та вихідними даними, а OnInit та OnDestroy - для відстеження подій життєвого циклу компонента.
* @Component({ ... }) Ця анотація визначає метадані компонента. У даному випадку, компонент має ім'я селектора 'app-post', шаблон визначений у файлі post.component.html, а стилі у файлі post.component.css. Компонент також реалізує інтерфейси OnInit та OnDestroy, що означає, що він реалізує методи життєвого циклу ngOnInit та ngOnDestroy.
* export class PostComponent implements OnInit, OnDestroy { ... } Оголошення класу PostComponent, який реалізує інтерфейси OnInit та OnDestroy. Цей клас відповідає за функціональність компонента.
* @Input() myPost!: Post; Це декларація властивості myPost, яка приймає вхідні дані типу Post. Вхідні дані передаються в компонент ззовні та можуть бути використані в шаблоні компонента. Знак оклику (!) після оголошення змінної myPost вказує на те, що ця змінна не є строгою (non-null). В контексті TypeScript та Angular це означає, що змінна myPost може мати значення null або undefined, і компілятор не буде розглядати це як помилку.
* @Output() onRemove = new EventEmitter<number>(); Ця декларація створює вихідну подію onRemove типу EventEmitter<number>. Ця подія буде використовуватися для сповіщення батьківського компонента про видалення посту.
* constructor() { ... } Конструктор класу PostComponent, в якому відбувається ініціалізація компонента. У цьому випадку, конструктор порожній.
* removePost() { ... } Цей метод викликається при натисканні на кнопку видалення посту. Він викликає emit на вихідній події onRemove і передає ідентифікатор посту для видалення.
* ngOnInit(): void { ... } Цей метод визначений інтерфейсом OnInit і викликається при ініціалізації компонента. У даному випадку, метод порожній.
* ngOnDestroy() { ... } Цей метод визначений інтерфейсом OnDestroy і викликається перед знищенням компонента. У цьому методі виводиться повідомлення у консоль.

Цей Angular компонент призначений для відображення постів та видалення їх за допомогою вхідних та вихідних даних, а також відстеження життєвого циклу компонента.

Тепер подивимося на файл post.component.html:

1. <div class="card">: Це HTML-елемент <div>, який має клас "card". Він служить контейнером для відображення всієї інформації про пост. Зазвичай такі класи використовуються для стилізації та форматування вмісту.
2. <h2>{{myPost.title}}</h2>: Цей HTML-елемент <h2> відображає заголовок посту, який береться з властивості myPost.title компонента. Значення myPost.title вставляється в текст заголовка за допомогою фігурних дужок {{}}, це називається інтерполяцією, і Angular автоматично замінює ці дужки значенням myPost.title.
3. <p>{{myPost.text}}</p>: Цей HTML-елемент <p> відображає текст посту, який береться з властивості myPost.text компонента. Також використовується інтерполяція для вставки значення myPost.text.
4. <button class="btn" (click)="removePost()">&times;</button>: Це HTML-елемент <button>, який має клас "btn". Цей елемент створює кнопку для видалення посту. Властивість (click)="removePost()" вказує на обробник події кліку, який викликає метод removePost() компонента при натисканні кнопки. Символ &times; вставлений в кнопку, зазвичай використовується для позначення опції видалення або закриття.
5. <ng-content></ng-content>: Цей тег <ng-content> використовується для включення контенту, який буде переданий в компонент через місце включення (<ng-content></ng-content>) в темплейті. Це дозволяє вставляти інший контент всередину компонента PostComponent при його використанні.

# **Детальний огляд компоненту post-form**

Розглаянемо файл post-form.component.ts:

Цей фрагмент коду представляє Angular компонент PostFormComponent, який відповідає за введення нового посту і додавання його до списку постів. Розглянемо кожен компонент і деталі його функціональності:

* import { Component, EventEmitter, OnInit, Output } from '@angular/core'; Ця рядок імпортує необхідні класи та функції з модуля @angular/core. Важливими є імпорт EventEmitter, який використовується для створення вихідних подій, та імпорт Post з app.component, що містить тип даних для постів.
* @Component({ ... }) Ця анотація визначає метадані компонента PostFormComponent. Вона визначає селектор 'app-post-form', шаблон і стилі компонента.
* export class PostFormComponent implements OnInit { ... } Оголошення класу PostFormComponent, який реалізує інтерфейс OnInit. Цей клас відповідає за функціональність компонента.
* @Output() onAdd: EventEmitter<Post> = new EventEmitter<Post>(); Ця декларація створює вихідну подію onAdd типу EventEmitter<Post>. Ця подія буде використовуватися для сповіщення батьківського компонента про додавання нового посту. Вихідна подія передає об'єкт типу Post.
* title = ''; Це поле містить заголовок нового посту. Воно ініціалізується рядком із порожнім значенням.
* text = ''; Це поле містить текст нового посту. Воно також ініціалізується рядком із порожнім значенням.
* constructor() { ... } Конструктор класу PostFormComponent, в якому відбувається ініціалізація компонента. У цьому випадку, конструктор порожній.
* ngOnInit(): void { ... } Цей метод визначений інтерфейсом OnInit і викликається при ініціалізації компонента. У даному випадку, метод порожній.
* addPost() { ... } Цей метод викликається при натисканні на кнопку "Додати". Він перевіряє, чи заповнені поля title та text, і якщо так, то створює новий об'єкт типу Post, викликає метод emit на вихідній події onAdd і передає новий пост для додавання. Після цього поля title та text очищаються, і в консоль виводиться повідомлення про додавання нового посту.

Цей Angular компонент дозволяє користувачу вводити нові пости (заголовок та текст) і додавати їх до списку постів шляхом виклику вихідної події onAdd.

Тепер подивимося на файл post-form.component.html:

* <input type="text" class="form-control" placeholder="Title..." [(ngModel)]="title">: Це текстове поле для введення заголовку нового посту. Важливо зауважити, що використовується директива [(ngModel)], яка забезпечує двостороннє зв'язування даних. Тобто, дані в полі вводу автоматично оновлюють значення властивості title у компоненті, і навпаки. Клас "form-control" використовується для стилізації поля вводу.
* <input type="text" class="form-control" placeholder="Text..." [(ngModel)]="text">: Це текстове поле для введення тексту нового посту. Як і в попередньому полі вводу, використовується двостороннє зв'язування даних за допомогою директиви [(ngModel)]. Клас "form-control" також використовується для стилізації поля вводу.
* <button class="btn" (click)="addPost()">Додати пост</button>: Це кнопка для додавання нового посту. Вона має клас "btn" для стилізації. При натисканні на цю кнопку викликається метод addPost(), який визначений у компоненті PostFormComponent. Тобто, при кліку на кнопку, дані, які були введені у полях вводу для заголовку та тексту, передаються в метод addPost() для обробки та додавання нового посту.

# **Детальний огляд компоненту app:**

Розглянемо компонент app.component.ts:

Цей фрагмент коду представляє Angular компонент AppComponent, який відповідає за основний вміст сторінки вашого додатку. Давайте розглянемо кожен компонент та його функціональність:

* import { Component } from '@angular/core'; Ця рядок імпортує клас Component з модуля @angular/core, який використовується для створення Angular компонентів.
* export interface Post { ... } Цей код визначає інтерфейс Post, який описує структуру об'єктів постів. Кожен пост має поля title (заголовок), text (текст) і необов'язкове поле id (ідентифікатор).
* @Component({ ... }) Ця анотація визначає метадані компонента AppComponent, включаючи селектор, шаблон і файли стилів.
* export class AppComponent { ... } Оголошення класу AppComponent, який відповідає за логіку і відображення сторінки.
* title = 'BlogComponents'; Це поле містить заголовок вашого додатку, який відображається у шаблоні.
* posts: Post[] = [...] Це поле містить масив об'єктів типу Post, який представляє пости в вашому додатку. У цьому прикладі визначені два пости з відповідними полями title, text і id.
* updatePosts(post: Post) { ... } Цей метод викликається при додаванні нового посту. Він отримує об'єкт типу Post і додає його до початку масиву posts за допомогою методу unshift(). Потім відбувається виведення цього посту в консоль.
* deletePost(id: number) { ... } Цей метод викликається при видаленні посту за його ідентифікатором. Він використовує метод filter() для фільтрації масиву posts та видалення посту з вказаним id.

Розглянемо компонент app.component.html:

1. <div class="container">: Цей <div> елемент має клас "container" і служить контейнером для всього вмісту сторінки. Зазвичай такі класи використовуються для стилізації та форматування вмісту сторінки.
2. <app-post-form (onAdd)="updatePosts($event)"></app-post-form>: Цей елемент включає компонент PostFormComponent, який представляє форму для створення нових постів. Директива (onAdd)="updatePosts($event)" визначає обробник події onAdd, який викликає метод updatePosts($event) в компоненті AppComponent, коли відбувається подія "додати пост". Відповідно, об'єкт посту передається в метод updatePosts() для додавання в список постів.
3. <hr />: Ця горизонтальна лінія (<hr>) вставлена для розділення форми для створення постів від відображення існуючих постів.
4. <app-post \*ngFor="let p of posts" [myPost]="p" (onRemove)="deletePost($event)">: Цей елемент використовує директиву \*ngFor для ітерації через масив posts і відображення кожного посту. Для кожного посту створюється екземпляр компонента PostComponent. Властивість [myPost]="p" передає дані посту у властивість myPost компонента PostComponent. Директива (onRemove)="deletePost($event)" визначає обробник події onRemove, який викликає метод deletePost($event) в компоненті AppComponent, коли відбувається подія "видалити пост". Ідентифікатор посту передається в метод deletePost() для видалення посту.
5. <small \*ngIf="p.text.length>20; else short">Пост довгий</small>: Ця директива \*ngIf визначає, що якщо текст посту має більше 20 символів, то відображається текст "Пост довгий". В іншому випадку використовується директива else, яка вказує на використання альтернативного шаблону <ng-template #short>, де відображається текст "Пост короткий".
6. <ng-template #short>: Цей тег <ng-template> визначає альтернативний шаблон для вмісту "Пост короткий". Цей шаблон використовується, коли умова в \*ngIf не виконується (текст посту короткий).

# **Посилання на розгорнутий додаток**

Додаток розміщений за адресою: <https://trofymovip02laba4.web.app/>

# **Висновки**

В даному завданні був розглянутий додаток «Blog». У ньому ми розглянули роботу з Angular, реалізували просту модель даних для цього додатку, опрацювали двосторонню прив’язку даних на реальному проєкті та навчилися користуватися функціоналом не строгої змінної, передачі однієї компоненти в інше за допомогою ng-model, а також методами ngOnInit() та ngOnDestroy(). Також у проєкті були використанні EventEmmiter, які допомогли для сповіщення батьківського елементу для додавання нових даних.

# **Список літератури**

* + 1. Introduction to the Angular docs [Електронний ресурс] – Режим доступу до ресурсу: <https://angular.io/docs>.
    2. Angular. Angular tutorials [Електронний ресурс] / Angular – Режим доступу до ресурсу: <https://angular.io/tutorial>.
    3. The Angular Book [Електронний ресурс] – Режим доступу до ресурсу: <https://angular-book.dev/>.
    4. Daniel Schmitz. Angular 14 from Scratch / Daniel Schmitz., 2022.
    5. Denys Vuika. Developing with Angular / Denys Vuika., 2018.