

Лабораторна робота № 2. Створення компонентів. Взаємодія між компонентами.

Мета: навчитися створювати компоненти та налагоджувати взаємодію між батьківським і дочірніми компонентами

Завдання

- 1) Ознайомитися з матеріалом лекції № 2;
- 2) Виконати пункти 1-5 ходу роботи;
- 3) У звіті відобразити кроки 1-5 ходу роботи;
- 4) Надати відповіді на контрольні питання.

Хід роботи

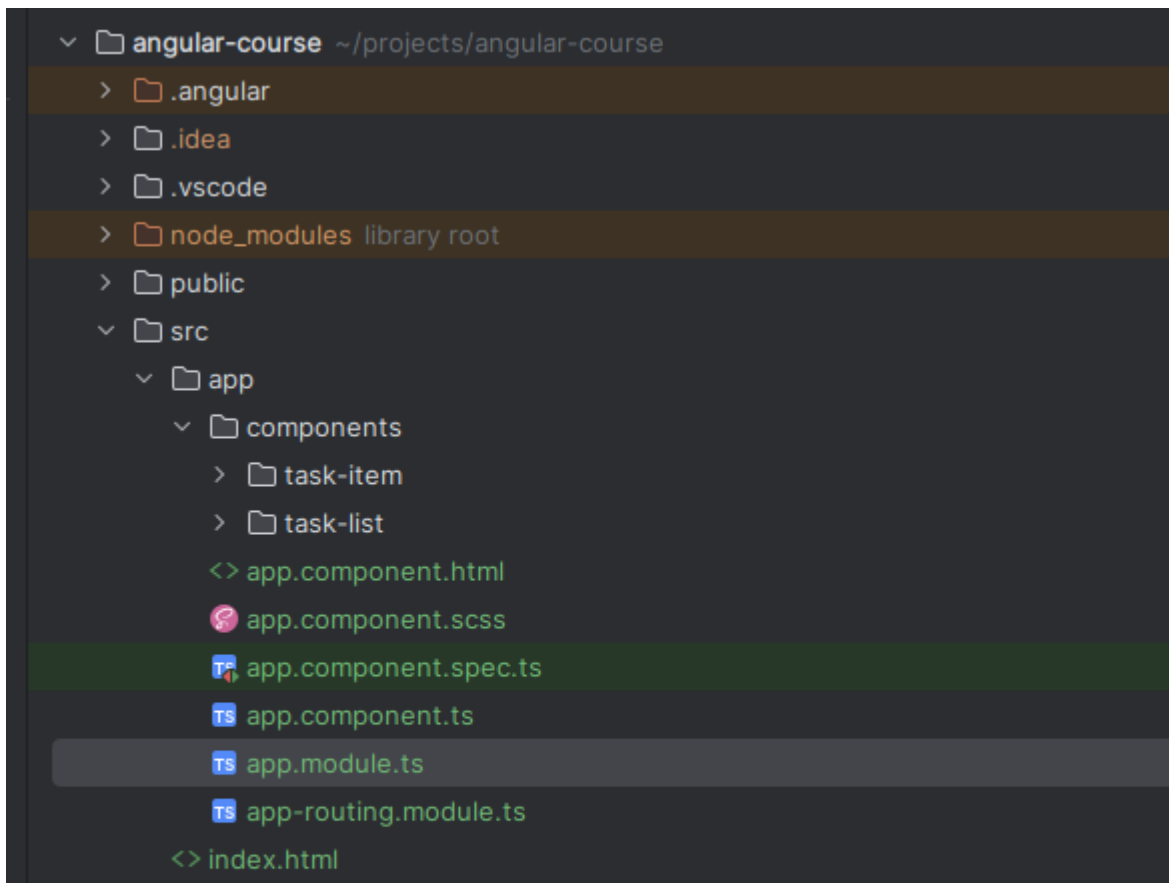
- 1) Створити два компонента:
 - a) TaskListComponent (батьківський компонент) - буде містити масив тасків та відклик на подію taskDeleted від TaskItemComponent;
 - b) TaskItemComponent (дочірній компонент) - буде приймати таск від батьківського компонента через @Input() і передає подію про його видалення (taskDeleted) через @Output().

```
Terminal Local x + v
svarog@svarog:~/projects/angular-course$ ng g c components/task-list
CREATE src/app/components/task-list/task-list.component.scss (0 bytes)
CREATE src/app/components/task-list/task-list.component.html (24 bytes)
CREATE src/app/components/task-list/task-list.component.spec.ts (612 bytes)
CREATE src/app/components/task-list/task-list.component.ts (232 bytes)
UPDATE src/app/app.module.ts (496 bytes)
svarog@svarog:~/projects/angular-course$ ng g c components/task-item
CREATE src/app/components/task-item/task-item.component.scss (0 bytes)
CREATE src/app/components/task-item/task-item.component.html (24 bytes)
CREATE src/app/components/task-item/task-item.component.spec.ts (612 bytes)
CREATE src/app/components/task-item/task-item.component.ts (232 bytes)
UPDATE src/app/app.module.ts (599 bytes)
svarog@svarog:~/projects/angular-course$
```

Після виконання команд, фреймворк автоматично запише створені компоненти у масив `declarations[]` файла `app.module.ts`

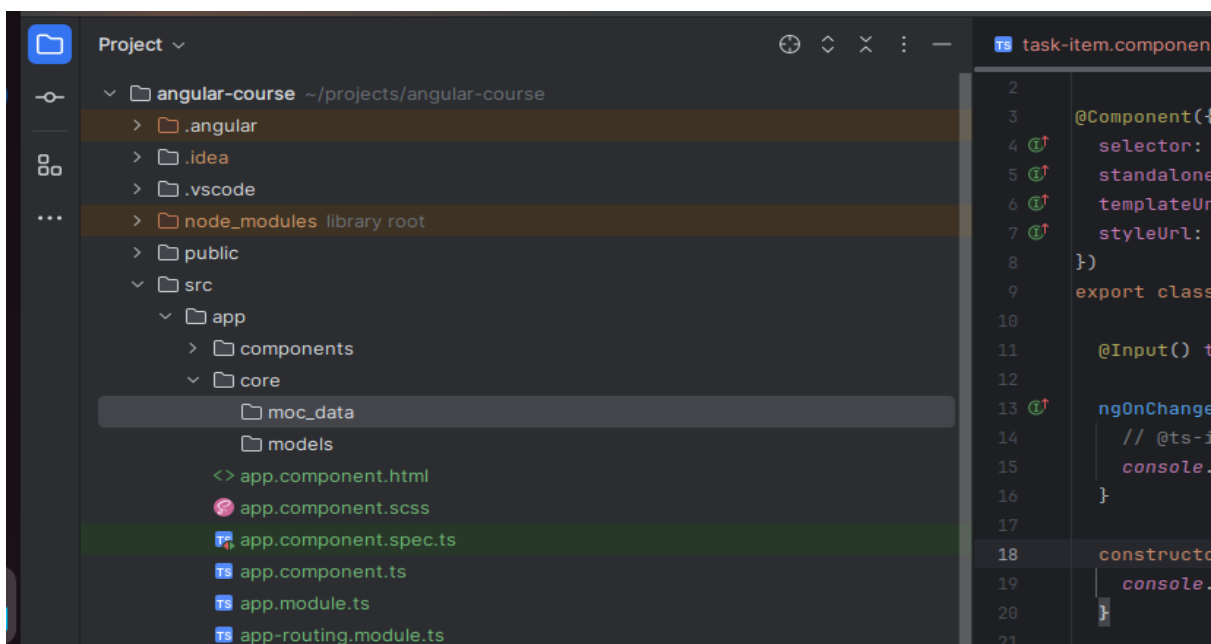
```
m+ README.md app.module.ts x
2 import { BrowserModule } from '@angular/platform-browser';
3
4 import { AppRoutingModule } from './app-routing.module';
5 import { AppComponent } from './app.component';
6 import { TaskListComponent } from './components/task-list/task-list.component';
7 import { TaskItemComponent } from './components/task-item/task-item.component';
8
9 @NgModule({ Show usages: new *
10   declarations: [
11     AppComponent,
12     TaskListComponent,
13     TaskItemComponent
14   ],
15   imports: [
16     BrowserModule,
17     AppRoutingModule
18   ],
19   providers: [],
20   bootstrap: [AppComponent]
21 })
22 export class AppModule { }
23
```

Новостворені компоненти можна знайти за шляхом `your_project/src/app/components`

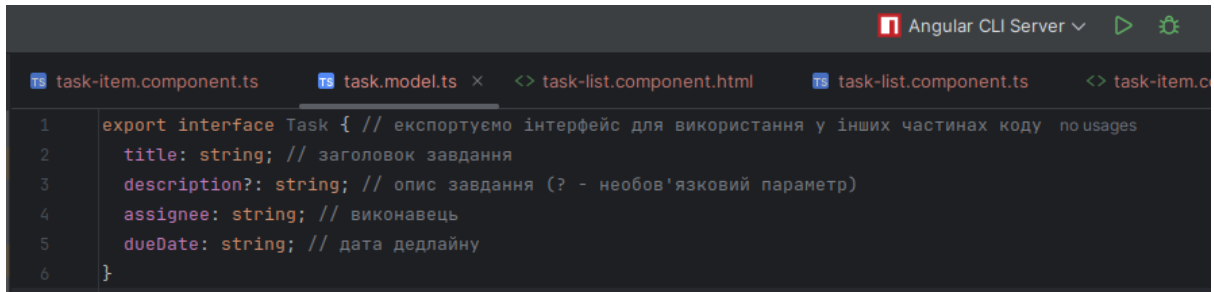


2) Створюємо теку **core** у теці **app**, у створеній теці створюємо дві теки:

- a) **models** - буде зберігати інтерфейси;
- b) **mock_data** - буде зберігати тестові набори даних.



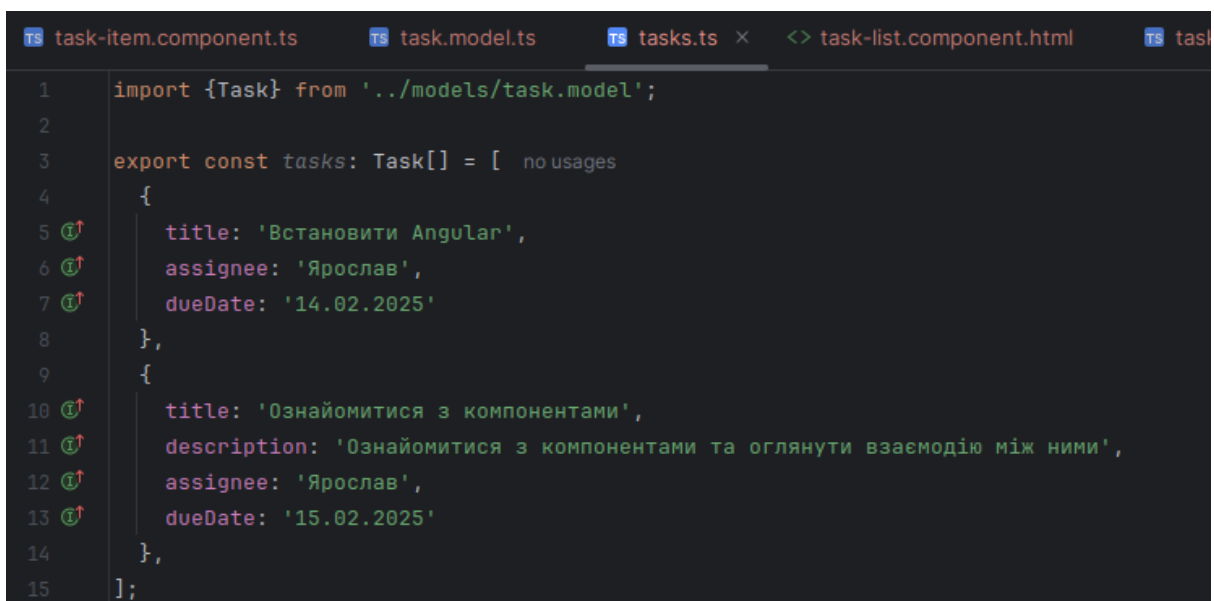
- 3) У теці **models** створюємо файл **task.model.ts** та прописуємо наступний код:



```
1 export interface Task { // експортуємо інтерфейс для використання у інших частинах коду no usages
2   title: string; // заголовок завдання
3   description?: string; // опис завдання (? - необов'язковий параметр)
4   assignee: string; // виконавець
5   dueDate: string; // дата дедлайну
6 }
```

Цей інтерфейс ми будемо використовувати як тип даних для змінних, що містять завдання.

- 4) У теці **mock_data** створюємо файл **tasks.ts** з наступним вмістом:



```
1 import {Task} from '../models/task.model';
2
3 export const tasks: Task[] = [ no usages
4   {
5     title: 'Встановити Angular',
6     assignee: 'Ярослав',
7     dueDate: '14.02.2025'
8   },
9   {
10    title: 'Ознайомитися з компонентами',
11    description: 'Ознайомитися з компонентами та оглянути взаємодію між ними',
12    assignee: 'Ярослав',
13    dueDate: '15.02.2025'
14  },
15 ];
```

- 5) Переходимо до реалізації компонентів, спочатку реалізуємо батьківський компонент, для цього виконаємо наступні дії:

- а) Замінімо стандартний html у файлі **app.component.html** на виклик тегу нашого батьківського компонента (він вказаний у властивості selector нашого файлу **task-list.component.ts**)

(оператор **!** після **task** каже TS, що змінна **task** буде ініціалізована пізніше, і він не повинен вважати її потенційно **null** або **undefined**)

```
task.model.ts tasks.ts task-list.component.html task-item.component.ts task-list.component.ts app.component.html
1 import {Component, Input} from '@angular/core';
2 import { Task } from '../core/models/task.model';
3
4 @Component({ Show usages
5   selector: 'app-task-item',
6   standalone: false,
7   templateUrl: './task-item.component.html',
8   styleUrls: ['./task-item.component.scss']
9 })
10 export class TaskItemComponent {
11   @Input() task!: Task; // отримуємо об'єкт завдання
12 }
```

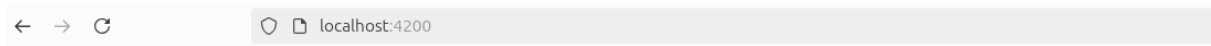
е) Далі прописуємо код, який буде відповідати за відображення у файлі **task-item.component.html**

```
task-item.component.ts task-list.component.html task-list.component.ts app.component.html task-item.component.html
1 <div class="task-item">
2   <h3>{{ task.title }}</h3> <!-- виводимо заголовок звертаючись то властивості title об'єкта tasks -->
3   <p *ngIf="task.description"><strong>Опис:</strong> {{ task.description }}</p>
4   <!-- виводимо опис звертаючись то властивості description об'єкта tasks, якщо вона є -->
5   <p><strong>Виконавець:</strong> {{ task.assignee }}</p>
6   <!-- виводимо виконавця звертаючись то властивості assignee об'єкта tasks -->
7   <p><strong>Дата виконання:</strong> {{ task.dueDate }}</p>
8   <!-- виводимо дату виконання звертаючись то властивості dueDate об'єкта tasks -->
9 </div>
10
```

Опціонально прописуємо стилі у файлі **task-item.component.scss**

```
task-item.component.ts task-item.component.scss task-list.component.html task-list.component.ts app.component.html
Enable File Watcher to compile SCSS to CSS? Ye
1 .task-item {
2   border: 1px solid #ccc;
3   padding: 15px;
4   border-radius: 5px;
5   margin-bottom: 10px;
6   background: #f9f9f9;
7 }
8
9 h3 {
10   margin: 0;
11   color: #333;
12 }
```

Отримали результат



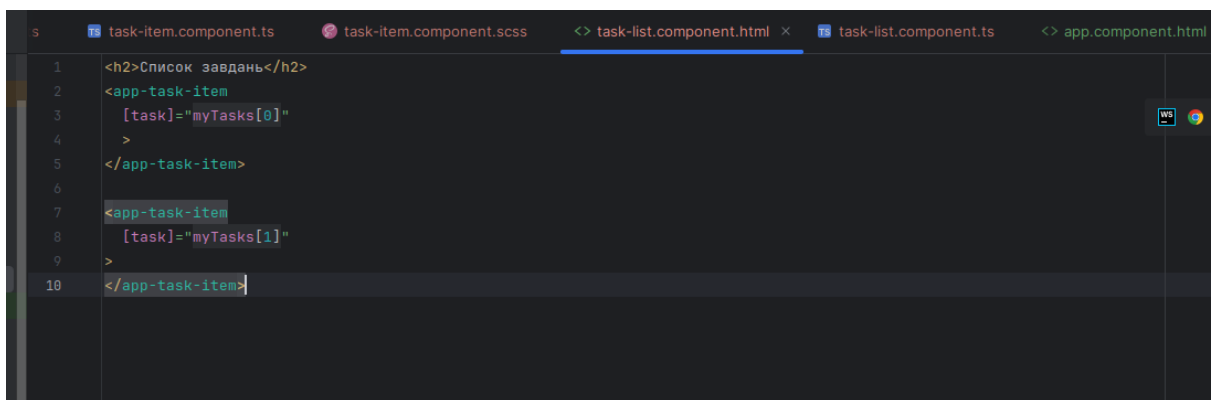
Список завдань

Встановити Angular

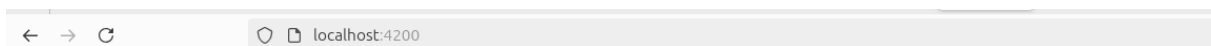
Виконавець: Ярослав

Дата виконання: 14.02.2025

Додамо ще одне завдання



Отримуємо



Список завдань

Встановити Angular

Виконавець: Ярослав

Дата виконання: 14.02.2025

Ознайомитися з компонентами

Опис: Ознайомитися з компонентами та оглянути взаємодію між ними

Виконавець: Ярослав

Дата виконання: 15.02.2025

Тепер зробимо зворотній потік даних за допомогою **@Output** на прикладі видалення завдання

- а) Для початку до файлу **task-item.component.html** додамо кнопку у якій пропишемо відгук на подію **click** назначивши її обробником метод **deleteTask()**

```
1 <div class="task-item">
2   <h3>{{ task.title }}</h3> <!-- виводимо заголовок звертаючись то властивості title об'єкта tasks -->
3   <p *ngIf="task.description"><strong>Опис:</strong> {{ task.description }}</p>
4   <!-- виводимо опис звертаючись то властивості description об'єкта tasks, якщо вона є -->
5   <p><strong>Виконавець:</strong> {{ task.assignee }}</p>
6   <!-- виводимо виконавця звертаючись то властивості assignee об'єкта tasks -->
7   <p><strong>Дата виконання:</strong> {{ task.dueDate }}</p>
8   <!-- виводимо дату виконання звертаючись то властивості dueDate об'єкта tasks -->
9   <button (click)="deleteTask()">Видалити</button>
10 </div>
11
```

- б) У файлі **task-item.component.ts** прописуємо метод **deleteTask()**

```
1 import {Component, Input} from '@angular/core';
2 import { Task } from '../core/models/task.model';
3
4 @Component({ Show usages
5   selector: 'app-task-item',
6   standalone: false,
7   templateUrl: './task-item.component.html',
8   styleUrls: ['./task-item.component.scss']
9 })
10 export class TaskItemComponent {
11   @Input() task!: Task; // отримуємо об'єкт завдання
12
13   deleteTask(): void { Show usages
14
15   }
16 }
17
```

Реалізуємо його за допомогою **@Output**

```
tasks.ts task-item.component.ts x task-item.component.scss <> task-list.component.html task-list.component.ts app.module.ts
1 import {Component, EventEmitter, Input, Output} from '@angular/core';
2 import { Task } from '../../core/models/task.model';
3
4 @Component({ Show usages
5   selector: 'app-task-item',
6   standalone: false,
7   templateUrl: './task-item.component.html',
8   styleUrls: ['./task-item.component.scss']
9 })
10 export class TaskItemComponent {
11   @Input() task!: Task; // отримуємо об'єкт завдання
12   @Output() taskDeleted: EventEmitter<void> = new EventEmitter<void>(); // подія для видалення завдання
13
14   deleteTask(): void { Show usages
15     this.taskDeleted.emit(); // викликаємо подію при натисканні кнопки "Видалити"
16   }
17 }
18
```

с) У файлі **task-list.component.html** прописуємо виклик кастомної події для кожного дочірнього елементу

```
tasks.ts task-item.component.ts task-item.component.scss <> task-list.component.html x task-list.component.ts app.module.ts
1 <h2>Список завдань</h2>
2 <app-task-item
3   [task]="myTasks[0]"
4   (taskDeleted)="deleteTask( number: 0)">
5 >
6 </app-task-item>
7
8 <app-task-item
9   [task]="myTasks[1]"
10   (taskDeleted)="deleteTask( number: 1)">
11 >
12 </app-task-item>
13 |
```

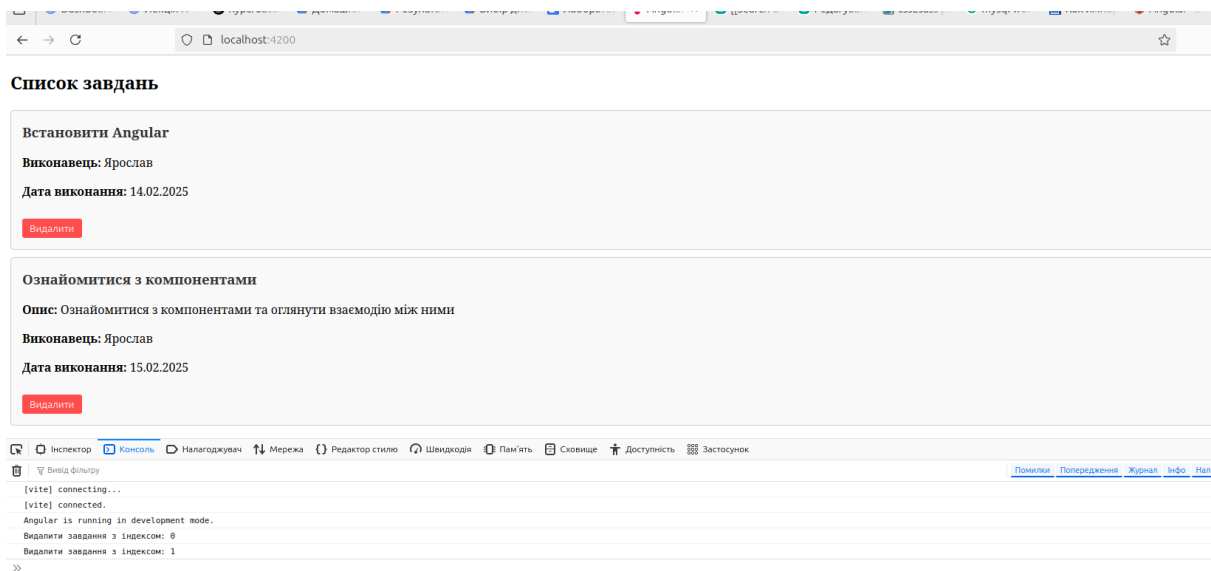
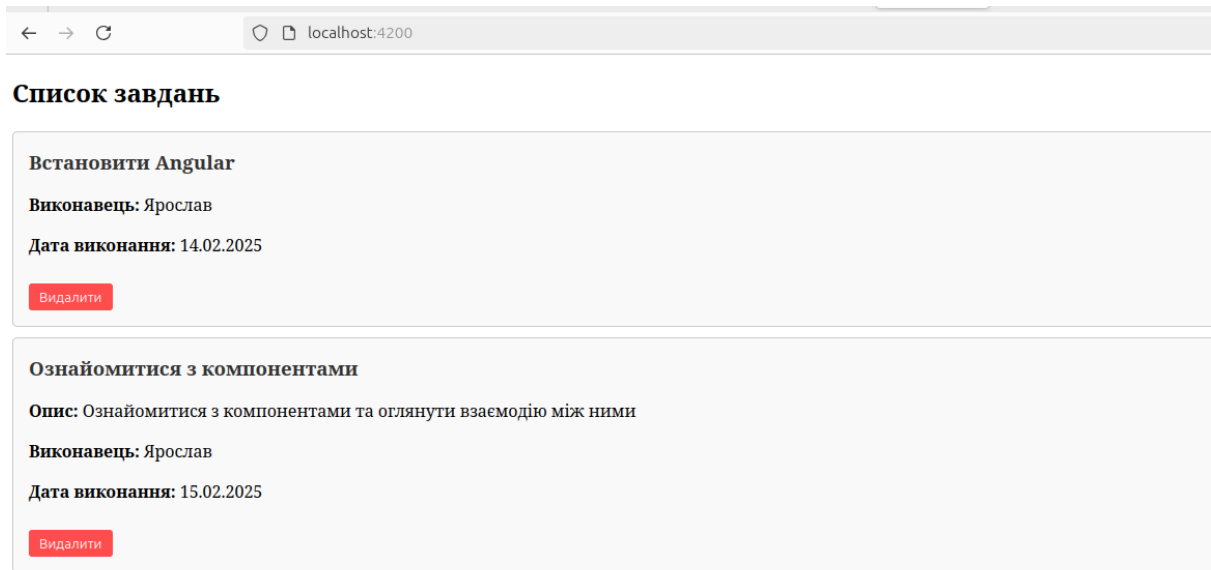
д) У файлі **task-list.component.ts** реалізуємо наш метод, який обробляє кастомну подію (емулюємо видалення)

```
tasks.ts task-item.component.ts task-item.component.scss task-list.component.html task-list.component.ts app.module.ts
1 import { Component } from '@angular/core';
2 import { Task } from '../../../core/models/task.model';
3 import { tasks } from '../../../core/moc_data/tasks';
4
5 @Component({ Show usages
6   selector: 'app-task-list',
7   standalone: false,
8   templateUrl: './task-list.component.html',
9   styleUrls: ['./task-list.component.scss']
10 })
11 export class TaskListComponent {
12
13   myTasks: Task[] = tasks; // підключаємо тестові дані, вказавши за тип наш інтерфейс, так як записів
14   // декілька, то вказуємо, що це буде масив
15
16   deleteTask(index: number): void { Show usages
17     console.log('Видалити завдання з індексом: ${index}');
18   }
19 }
20
```

Опціонально стилізуємо кнопку видалення

```
tasks.ts task-item.component.ts task-item.component.scss task-list.component.html task-list.component.ts app.module.ts
Enable File Watcher to compile SCSS to CSS? Yes
1 .task-item {
2
3   border-radius: 5px;
4   margin-bottom: 10px;
5   background: #f9f9f9;
6 }
7
8
9 h3 {
10   margin: 0;
11   color: #333;
12 }
13
14 button {
15   background-color: #ff4d4d;
16   color: white;
17   border: none;
18   padding: 5px 10px;
19   cursor: pointer;
20   border-radius: 3px;
21   margin-top: 10px;
22
23   &:hover {
24     background-color: #cc0000;
25   }
26 }
27
```

Отримуємо наступний результат



Також можна передавати відповідні параметри, при виклику кастомної події. Для цього внесемо певні корективи у наш код

У файл **task.model.ts** додамо значення **id**

```
task.model.ts x tasks.ts task-item.component.ts task-item.component.scss task-list.component.html t
1 export interface Task { // експортуємо інтерфейс для використання у інших частинах коду Show usages
2   id: number;
3   title: string; // заголовок завдання
4   description?: string; // опис завдання (? - необов'язковий параметр)
5   assignee: string; // виконавець
6   dueDate: string; // дата дедлайну
7 }
8
```

Модернізуємо файл **tasks.ts**

```
task.model.ts tasks.ts x task-item.component.ts task-item.component.scss task-list.component.html task-list.compo
1 import {Task} from '../models/task.model';
2
3 export const tasks: Task[] = [ Show usages
4   {
5     id: 0,
6     title: 'Встановити Angular',
7     assignee: 'Ярослав',
8     dueDate: '14.02.2025'
9   },
10  {
11    id: 1,
12    title: 'Ознайомитися з компонентами',
13    description: 'Ознайомитися з компонентами та оглянути взаємодію між ними',
14    assignee: 'Ярослав',
15    dueDate: '15.02.2025'
16  },
17 ];
```

Пропишемо зміни у **task-item.component.html** додавши параметр **task.id** для методу

```
task.model.ts tasks.ts task-item.component.ts task-item.component.html x task-item.component.scss task-list.component
1 <div class="task-item">
2   <h3>{{ task.title }}</h3> <!-- виводимо заголовок звертаючись то властивості title об'єкта tasks -->
3   <p *ngIf="task.description"><strong>Опис:</strong> {{ task.description }}</p>
4   <!-- виводимо опис звертаючись то властивості description об'єкта tasks, якщо вона є -->
5   <p><strong>Виконавець:</strong> {{ task.assignee }}</p>
6   <!-- виводимо виконавця звертаючись то властивості assignee об'єкта tasks -->
7   <p><strong>Дата виконання:</strong> {{ task.dueDate }}</p>
8   <!-- виводимо дату виконання звертаючись то властивості dueDate об'єкта tasks -->
9   <button (click)="deleteTask(task.id)">Видалити</button>
10 </div>
11
```

Внесемо зміни у **task-item.component.ts** змінивши тип даних для ***EventEmitter*** та передавши параметр **id** у ***this.taskDeleted.emit***

```
task.model.ts | tasks.ts | task-item.component.ts x | <> task-item.component.html | task-item.component.scss | <> task-list.component.html v
1 import {Component, EventEmitter, Input, Output} from '@angular/core';
2 import { Task } from '../../core/models/task.model';
3
4 @Component({ Show usages
5   selector: 'app-task-item',
6   standalone: false,
7   templateUrl: './task-item.component.html',
8   styleUrls: ['./task-item.component.scss']
9 })
10 export class TaskItemComponent {
11   @Input() task!: Task; // отримуємо об'єкт завдання
12   @Output() taskDeleted: EventEmitter<number> = new EventEmitter<number>(); // подія для видалення завдання
13
14   deleteTask(id: number): void { Show usages
15     this.taskDeleted.emit(id); // викликаємо подію при натисканні кнопки "Видалити"
16   }
17 }
```

Модернізуємо файл **task-list.component.html** додавши до параметрів функції **\$event**

```
task.model.ts | tasks.ts | task-item.component.ts | <> task-item.component.html | task-item.component.scss | <> task-list.component.html x v
1 <h2>Список завдань</h2>
2 <app-task-item
3   [task]="myTasks[0]"
4   (taskDeleted)="deleteTask($event)">
5   >
6 </app-task-item>
7
8 <app-task-item
9   [task]="myTasks[1]"
10   (taskDeleted)="deleteTask($event)">
11   >
12 </app-task-item>
13 |
```

Перевіряємо роботу застосунку

← → ↺

localhost:4200

Список завдань

Встановити Angular
Виконавець: Ярослав
Дата виконання: 14.02.2025

Видалити

Ознайомитися з компонентами
Опис: Ознайомитися з компонентами та оглянути взаємодію між ними
Виконавець: Ярослав
Дата виконання: 15.02.2025

Видалити

Інспектор

Консоль

Налагоджувач

Мережа

Редактор стилів

Швидкодія

Пам'ять

Сховище

Доступність

Застосунок

Вивід фільтру

[vite] connecting...

[vite] connected.

Angular is running in development mode.

Видалити завдання з індексом: 0

Видалити завдання з індексом: 1

>>

Контрольні питання

- 1) Як створити новий компонент?
- 2) Для чого використовується `@Input`?
- 3) Для чого використовується `@Output`?
- 4) Яка властивість відповідає за унікальний ідентифікатор компонента, який дозволяє використовувати його на інших сторінках?
- 5) Як вивести змінну у шаблоні?