**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

# «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки Кафедра інформатики та програмної інженерії

Звіт по лабораторній роботі № 4

Сервіси. Створення локальних та глобальних сервісів.

з дисципліни: «Реактивне програмування»

Студент: Головня Олександр Ростиславович Група: ІП-11 Дата захисту роботи: Викладач: доц. Полупан Юлія Вікторівна Захищено з оцінкою:

Київ, 2024

**Зміст**

[«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО» 1](#_Toc180898801)

[**Вправа 1:** 3](#_Toc180898802)

[**Вправа 2:** 7](#_Toc180898803)

[**Впровадження сервісу в інший сервіс** 7](#_Toc180898804)

[**Вправа 3:** 8](#_Toc180898805)

[**Опціональні сервіси** 8](#_Toc180898806)

[**Вправа 4:** 10](#_Toc180898807)

[**Один сервіс для всіх компонентів** 10](#_Toc180898808)

[**Ієрархія сервісів** 14](#_Toc180898809)

[**Кореневий рівень** 15](#_Toc180898810)

[**Рівень модуля** 16](#_Toc180898811)

[**Рівень компоненту** 17](#_Toc180898812)

[**Вправа 5:** **Виконати самостійно:** 17](#_Toc180898813)

[**Виконання** 17](#_Toc180898814)

[**Вправа 6: Виконати самостійно:** 21](#_Toc180898815)

[**Висновок:** 25](#_Toc180898816)

[**Список використаних джерел:** 26](#_Toc180898817)

**Мета: Мета: Навчитися створювати та використовувати сервіси у Angular.**

**Завдання: Завдання: Створити два Angular-додатки під назвою Service1 та Service2.**

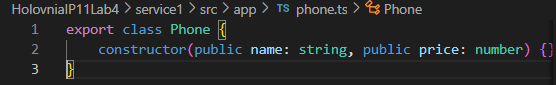
1. **Для Angular-додатку Service1 виконати вправи 1-4;**
2. **Для Angular-додатку Service2 виконати самостійно вправу 5.**
3. **Для Angular-додатку Service3 виконати самостійно вправу 6.**
4. **Зробити звіт по роботі.**
5. **Angular-додаток Service2 та Service3 розгорнути на платформі Firebase у проекті з ім’ям «ПрізвищеГрупаLaba4-2» та «ПрізвищеГрупаLaba4-3», наприклад «KovalenkoIP01Laba4-2» та «KovalenkoIP01Laba4-3».**

**Вправа 1:**

Допустимо, у нас є наступна базова структура проекту:

Додамо до папки src/app новий файл, який назвемо phone.ts. Визначимо у цьому

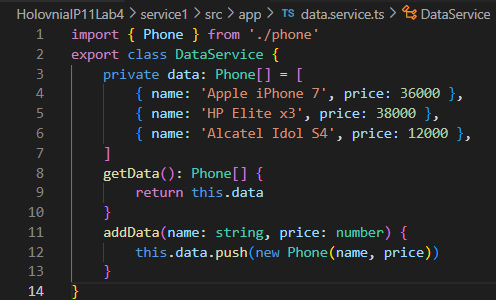
файлі наступний клас:



Цей клас представлятиме дані, з якими ми працюватимемо.

Далі додамо до папки src/app новий файл data.service.ts. Цей файл міститиме код сервісу. Згідно з прийнятими в Angular умовностями в назві файлу після назви самого сервісу має йти підрядок .service. Якщо назва сервісу складається з кількох слів, які починаються з великої літери, то у назві файлу всі ці слова пишуться з маленької літери та поділяються дефісами. Наприклад, якщо сервіс називається SpecialSuperHeroService, то файл сервісу буде мати назву special-super-hero.service.ts.

Визначимо у цьому файлі наступний сервіс:



У сервісі визначено масив даних та методи для роботи з ним. У реальному додатку ці дані можна отримувати з сервера або будь-якого зовнішнього сховища.

Тепер визначимо код компонента:





Щоб задіяти сервіс у компоненті, його не тільки треба імпортувати:

import { DataService } from './data.service'

Але також необхідно його додати до колекції providers компонента:

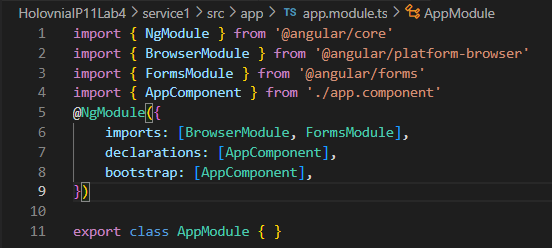
providers: [DataService]

Усі сервіси, що використовуються, повинні бути визначені в колекції providers. І після цього ми зможемо задіяти вбудований у Angular механізм dependency injection та отримати об'єкт сервісу в конструкторі компонента і потім використовувати за потребою:

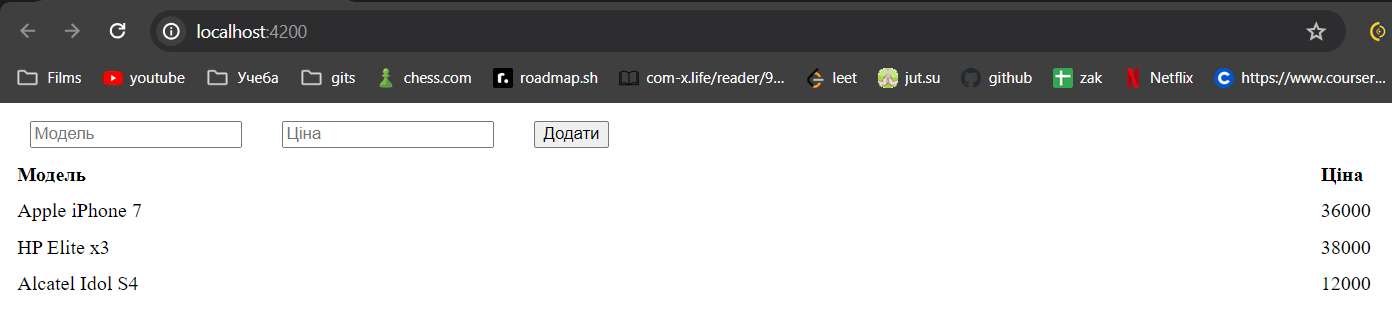
constructor(private dataService: DataService){}

Для завантаження даних реалізується метод ngOnInit() інтерфейсу OnInit, що викликається після конструктора.

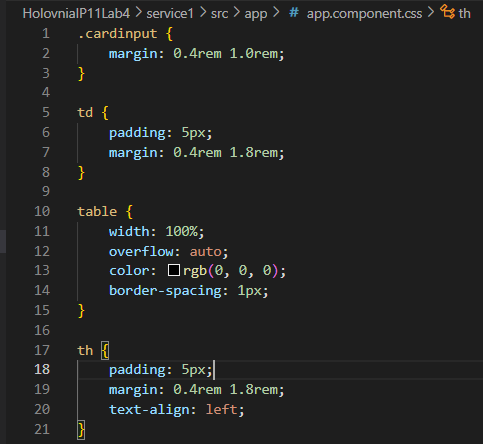
І тому що ми використовуємо директиву ngModel для роботи з елементами форми, нам потрібно підключити модуль FormsModule у файлі app.module.ts:



У результаті ми зможемо виводити дані та додавати нові елементи через сервіс:



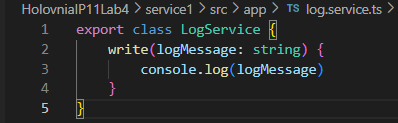
Файл стилів для компонента app.component.ts має наступний вміст:



**Вправа 2:**

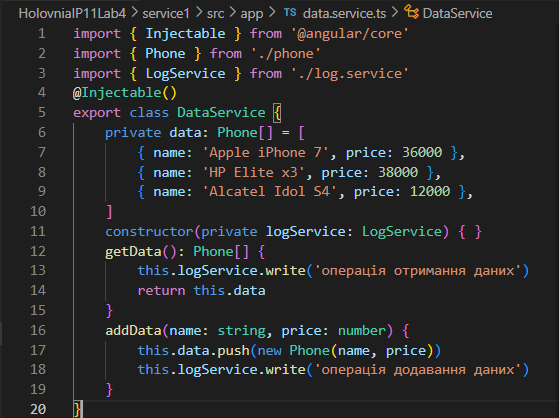
**Впровадження сервісу в інший сервіс**

Цілком ймовірна ситуація, коли ми захочемо використовувати один сервіс в іншому сервісі. Наприклад, у вправі 1 було створено сервіс для роботи з даними. Що якщо нам необхідно логувати всі операції з даними. Для логування визначимо новий сервіс. Для цього додамо до папки src/app новий файл log.service.ts з наступним вмістом:

****

Для логування у сервісі визначено метод write, який виводить повідомлення на консоль.

Тепер використовуємо цей сервіс. Для цього змінимо код у файлі data.service.ts:

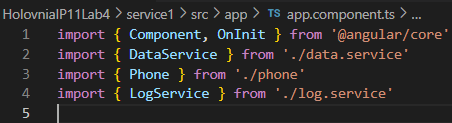


Щоб вказати, що сервіс може використовувати інші сервіси, до класу сервісу застосовується декоратор@Injectable. Якщо клас не матиме такого декоратора, то вбудований механізм застосування залежностей не зможе створити об'єкт цього класу і видасть помилку.

Існує загальна рекомендація від розробників Angular застосовувати @Injectable добудь-якого класу сервісу, хоча це необов'язково.

Хоча у вправі 1 ми могли використовувати сервіс у компоненті без застосування до компоненту декоратора @Injectable. Справа в тому, що декоратор@Component, який застосовується до компонента, є підтипом @Injectable.

І також у випадку з DataService сервіс LogService також треба зареєструвати у списку провайдерів AppComponent:

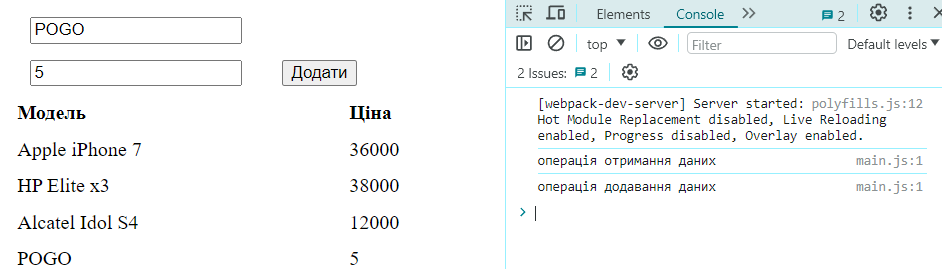




І незважаючи на те, що безпосередньо LogService не використовується в

компоненті AppComponent, але він використовується в DataService, який викликається в AppComponent.

І при виконанні операцій із даними в консолі браузера ми зможемо побачити роботу сервісу LogService.



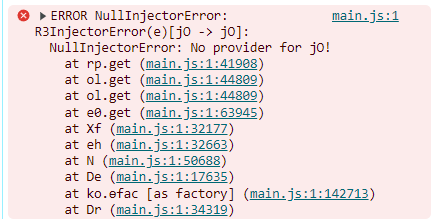
**Вправа 3:**

**Опціональні сервіси**

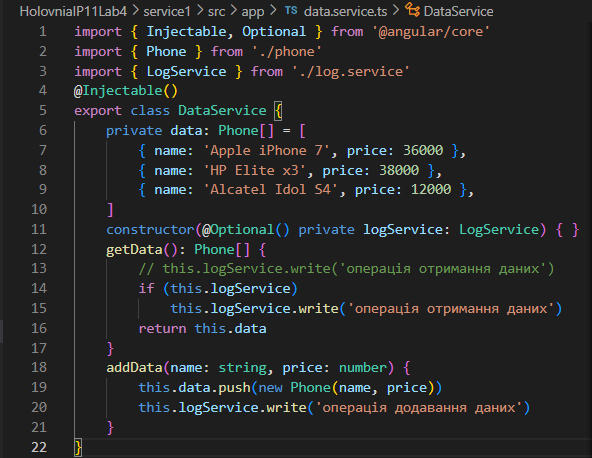
Припустимо, з якоїсь причини сервіс LogService не доступний для інжектування, наприклад ми не додали в провайдери компонента AppComponent. Наприклад, буде зазначено так:

****

Якщо ми запустимо програму, то в цьому випадку ми отримаємо помилку. Оскільки для сервісу LogService не визначено провайдера.

****

І тут можна визначити сервіс LogService як опціональний, застосовуючи декоратор Optional. Для цього змінимо код DataService:

****

Далі при зверненні до сервісу тепер потрібно перевірити, чи він встановлений, і

якщо він встановлений, використовувати його:

****

При цьому помилки вже не буде.

**Вправа 4:**

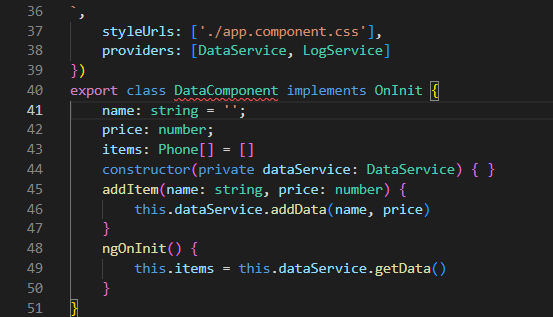
**Один сервіс для всіх компонентів**

Цілком можливо, що у нашому додатку буде кілька різних компонентів, які використовують сервіси. При цьому компоненти можуть використовувати ті самі сервіси. Наприклад, у попередніх вправах було створено сервіс DataService, який керує даними.

Визначимо спеціальний компонент для роботи з даними. Для цього візьмемо проект із минулої вправи та додамо до папки src/app новий файл data.component.ts:

Визначимо у цьому файлі наступний код:





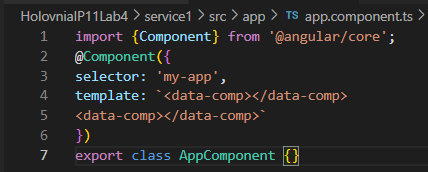
DataComponent завантажує та додає дані. Для роботи із сервісами декоратор

Component визначає секцію providers:

providers: [DataService, LogService]

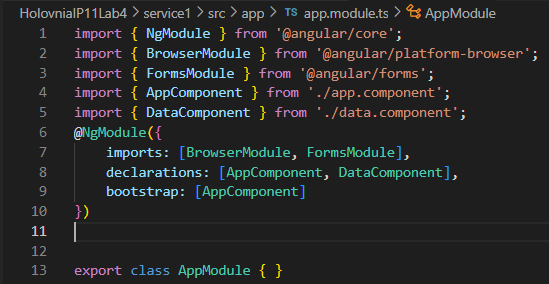
Використовуємо цей компонент DataComponent у головному компоненті

AppComponent:

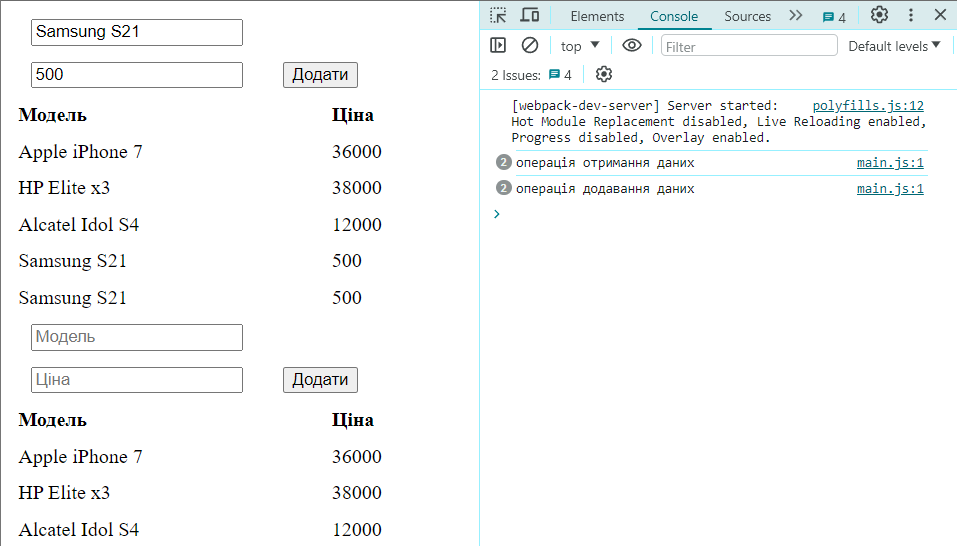


Цей компонент через елемент data-comp викликає компонент DataComponent. Причому викликає двічі. Тобто для обробки кожного елемента створюватиметься свій об'єкт DataComponent.

І відповідно змінимо головний модуль програми AppModule:



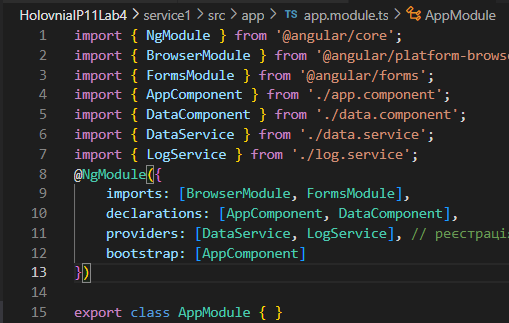
Тепер запустимо програму і спробуємо додати новий елемент:



Як видно на скріншоті, при додаванні в одному компоненті новий елемент

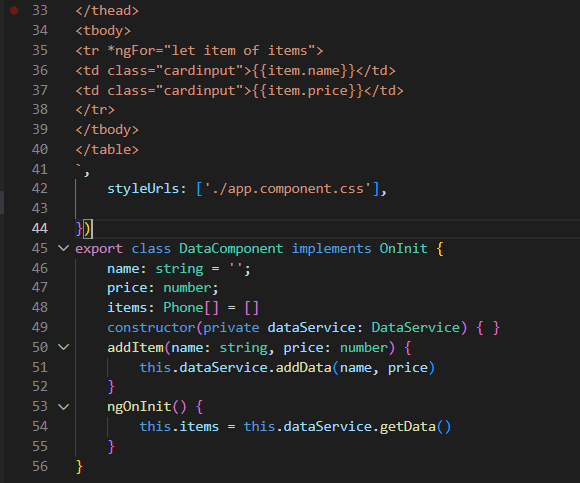
додаватиметься тільки для цього компонента. Тому що у нас два окремі компоненти, і для кожного з них буде створюватись свій набір сервісів DataService та LogService.

Така поведінка не завжди може бути кращою. Можливо, потрібно, щоб компоненти використовували той самий об'єкт сервісу, замість створення різних сервісів кожного компонента. Для цього ми можемо зареєструвати всі сервіси не в компоненті, а в головному модулі AppModule:



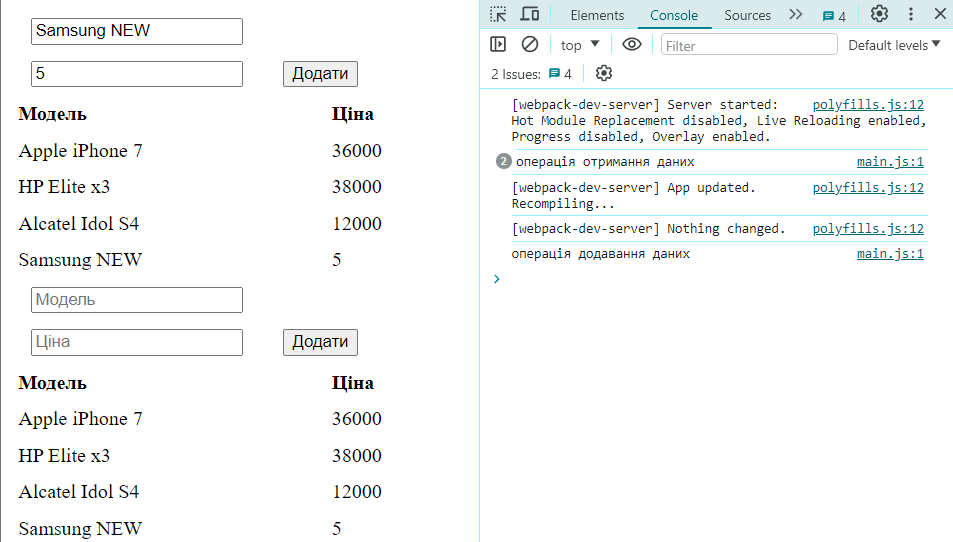
У цьому випадку ми вже можемо прибрати реєстрацію сервісів з DataComponent:





Після цього обидва об'єкти DataComponent будуть використовувати той самий

сервіс DataService. Тому додавання об'єкта в одному компоненті автоматично позначиться на іншому:



**Ієрархія сервісів**

Як було показано раніше, ми можемо впроваджувати сервіси у компонентах та модулях. Залежно від цього, де впроваджуються послуги, бувають різні рівні провайдерів сервісів. Рівень по суті визначає сферу дії та життєвий цикл сервісу. Є три рівні провайдерів сервісів:

1) Глобальний чи кореневий рівень

2) Рівень модуля

3) Рівень компоненту

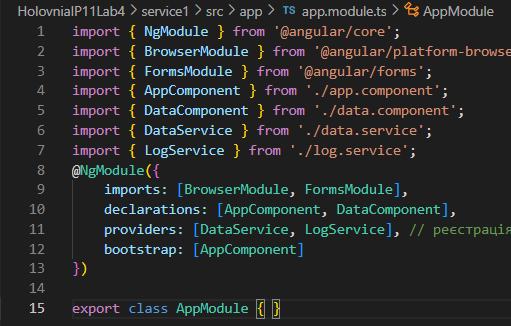
Для встановлення відповідного рівня сервісу є два способи: додавання сервісу до колекції providers у модулі або компоненті та встановлення рівня за допомогою параметра providedIn у декораторі @Injectable.

З одного боку, може здатися, що немає сенсу в такому розподілі - чому б для всіх сервісів не зробити один кореневий рівень, щоб один об'єкт сервісу був доступний для всього додатку на кшталт синглтона. Однак нерідко виникає необхідність розмежувати функціонал між окремими функціональними частинами додатка. Наприклад, коли два компоненти працюють із різним набором даних – в цьому випадку вони можуть використовувати один клас сервісу, але різні його об'єкти.

**Кореневий рівень**

Кореневий рівень (root level) передбачає дію сервісу для всієї програми.

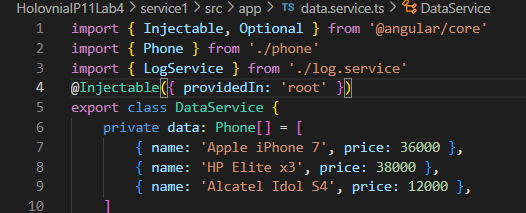
Створюється та використовується один об'єкт сервісу для всіх частин програми. Подібний рівень встановлюється, якщо сервіс додається до колекції providers головного модуля, який зазвичай називається AppModule і з якого починається робота програми. Наприклад:



Якщо AppModule є головним модулем, то для сервісу DataService визначено кореневий рівень. Тобто під час роботи програми буде створюватись один об'єкт даного сервісу для всієї програми.

Значення root для параметра providedIn у декораторі Injectable також дозволяє

встановити кореневий рівень дії:



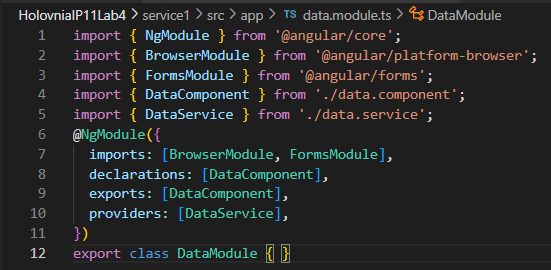
У цьому випадку ми можемо не додавати цей сервіс до колекції providers у

головному модулі.

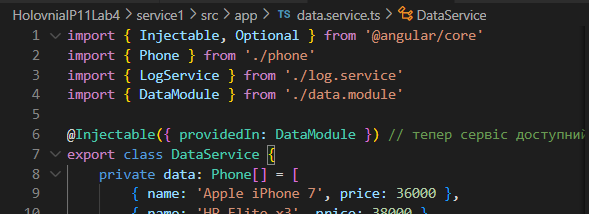
**Рівень модуля**

Сервіси рівня модуля діють лише для поточного модуля і його класів -

компонентів, директив, сервісів. Це все ті сервіси, які додаються до колекції providers у всіх інших модулях, крім головного модуля. Наприклад:



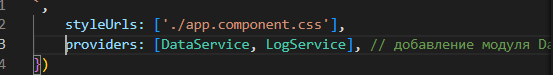
Або ми можемо задати рівень модуля безпосередньо у сервісі:



**Рівень компоненту**

І тут дія сервісу обмежена поточним компонентом. Для кожного об'єкта

компонента створюється об'єкт сервісу. Сам сервіс додається також до колекції providers компонента:

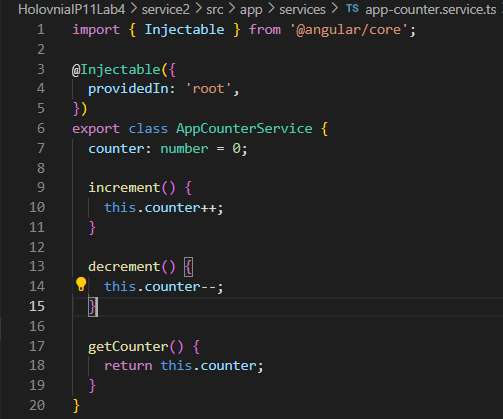


**Вправа 5:** **Виконати самостійно:**

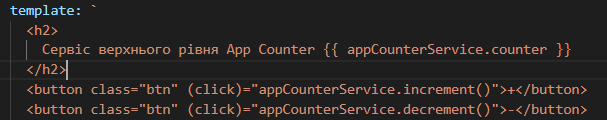
Створимо новий проект Service2. Створимо папку app/services. У ній два сервіси з різними областями видимості. Перший сервіс app-counter.service.ts глобального рівня. Інший сервіс local-counter.service.ts локального рівня. Використайте глобальний сервіс app-counter.service.ts та локальний сервіс local-counter.service.ts для двох компонентів app.component.ts та counter.component.ts як показано нижче (компонент counter.component.ts створіть самостійно):

**Виконання**

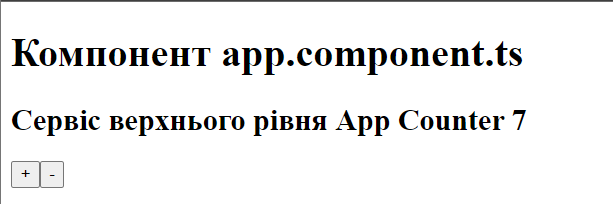
1) Перший сервіс app-counter.services з класом AppCounterService кореневого рівня, який зареєстрований глобально та інжектований у компоненті app.component.ts має наступний вміст:



Шаблон app.component.html при цьому має наступний вміст:



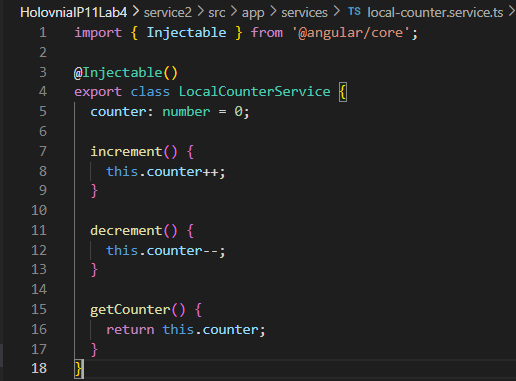
Результат:



При цьому кнопки "+" та "-" працюють зменшуючи або збільшуючи значення

counter.

2) Інший сервіс local-counter.services з класом LocalCounterService зареєстрований локально на рівні компоненту app.component.ts та інжектований в нього. Сервіс має наступний вміст:



Результат:



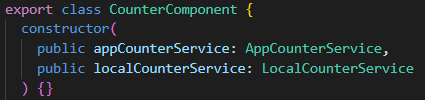
3) Створимо окремий компонент у папці app/counter з ім'ям counter.component.ts та імпортуйте в нього сервіс глобального рівня app-counter.services. У шаблоні цього компоненту повторіть логіку app.component.html.



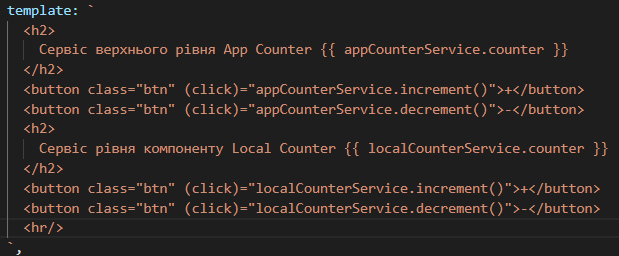
4) Окремо у властивості providers компонента counter.component.ts зареєструємо локальний сервіс LocalCounterService наступним чином:

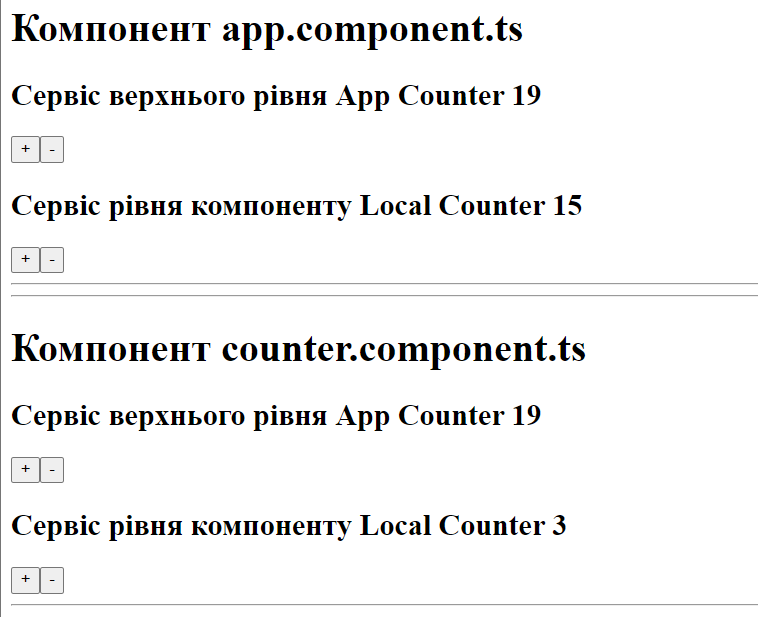


та інжектуємо його наступним чином:



Змінимо шаблон counter.component.html наступним чином:





**Вправа 6: Виконати самостійно:**

Створіть новий проект Service3, в якому створіть сервіс для отримання

користувачів (users) з сервера https://jsonplaceholder.typicode.com/users. Отримавши відповідь від сервера, вивести дані користувачів на сторінку в наступному виді:



(Для визуальної частини можно використати bootstrap або інші фреймворки).

При цьому завдання має два варіанти! 1 Варіант – парні номери в загальному

списку; 2 Варіант – непарні номери в загальному списку.



**1 Варіант: При виведенні даних, у шаблоні використовувати async пайп;**

2 Варіант: При виведенні даних, у шаблоні не використовувати async пайп.

Інсталюємо Bootstrap (опціонально для стилізації):

npm install bootstrap

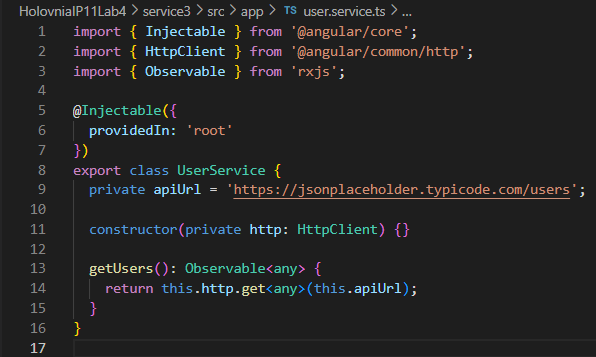
Додамо Bootstrap у файл angular.json:

"styles": [ "node\_modules/bootstrap/dist/css/bootstrap.min.css", "src/styles.css" ],

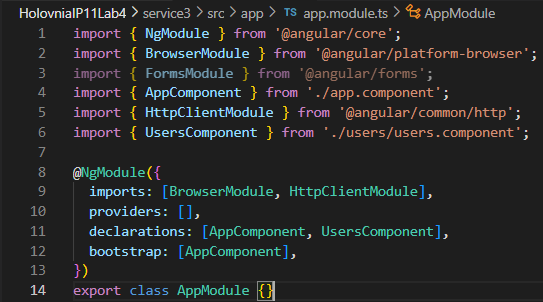
Створимо сервіс для отримання користувачів: Виконаємо команду:

ng generate service user

У файлі src/app/user.service.ts:



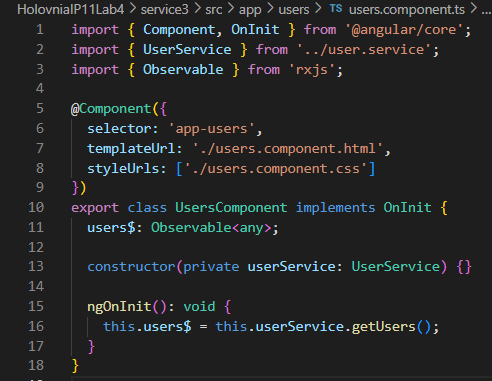
Імпортуємо HttpClientModule в src/app/app.module.ts:



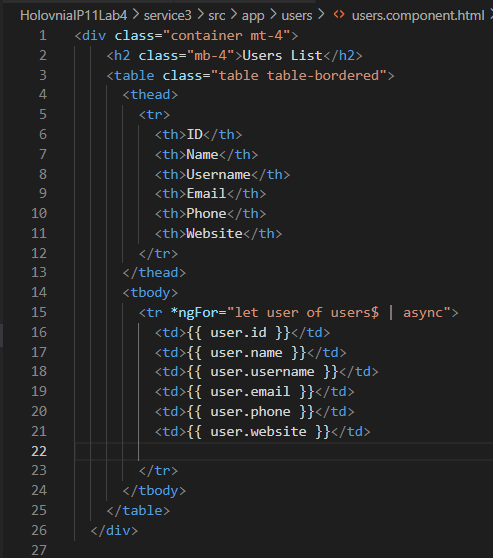
Створимо компонент для відображення користувачів: Виконаємо команду:

ng generate component users

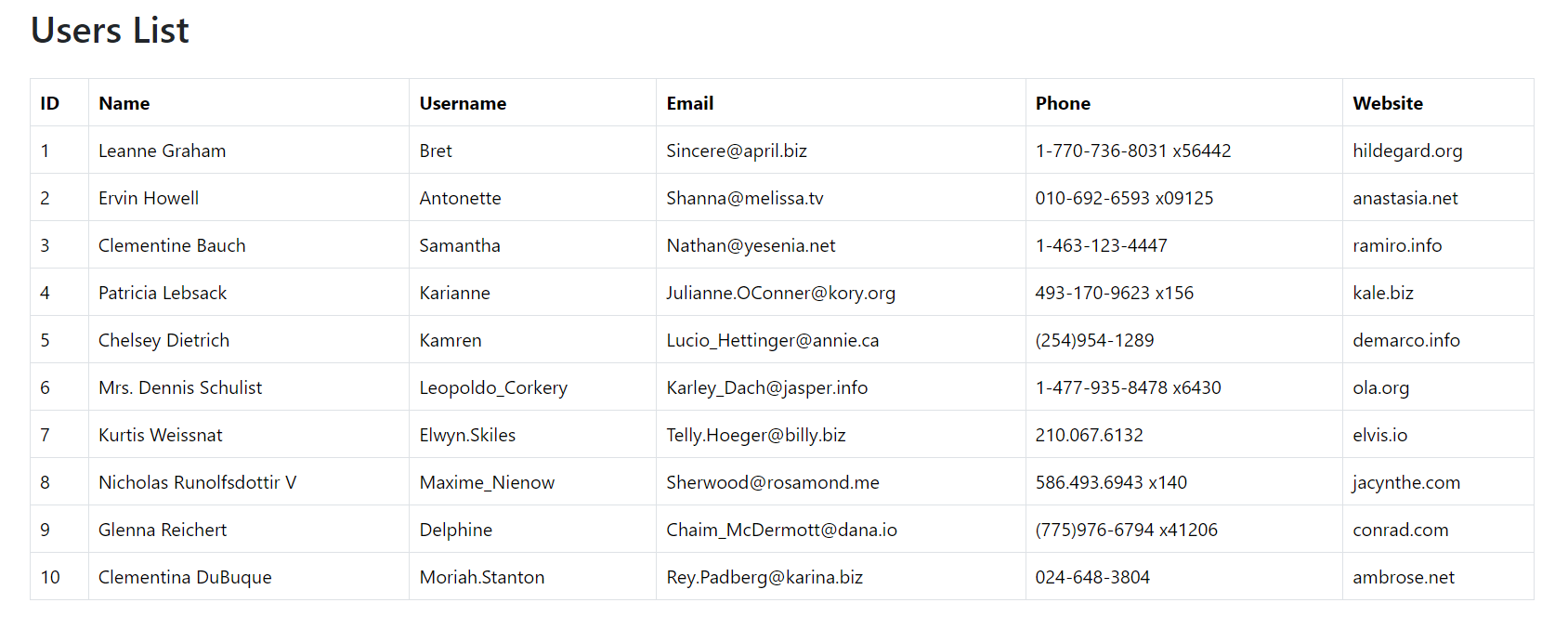
У компоненті src/app/users/users.component.ts:



У шаблоні src/app/users/users.component.html використовуємо async пайп для виведення користувачів:



Результат бачимо на рисунку:



Посилання на додатки:

<https://holovniaip11s2.web.app/>

<https://holovniaip11s3.web.app/>

**Висновок:**

Під час виконання комп’ютерного практикуму навчився створювати та використовувати сервіси у Angular. Дізнався про їх призначення та використання,

Мною було створено 3 додатка. Згідно завдання, розгорнув Angular-додатки «Service2» та «Service3» на платформі FireBase.

**Список використаних джерел:**

1. Understanding Services: <https://v17.angular.io/guide/creating-injectable-service>