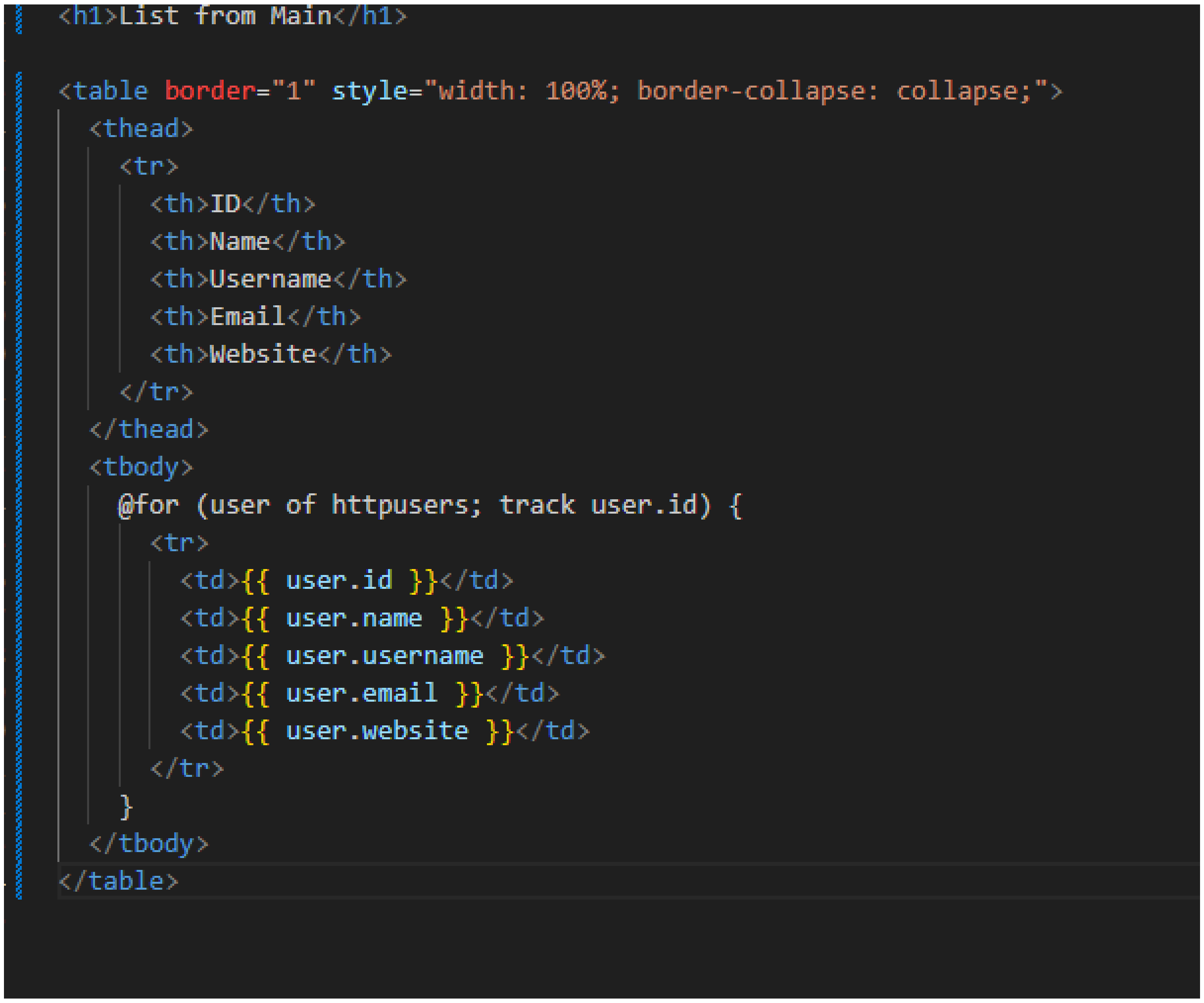
Pereda Jan Mark S. WD-304

App.html

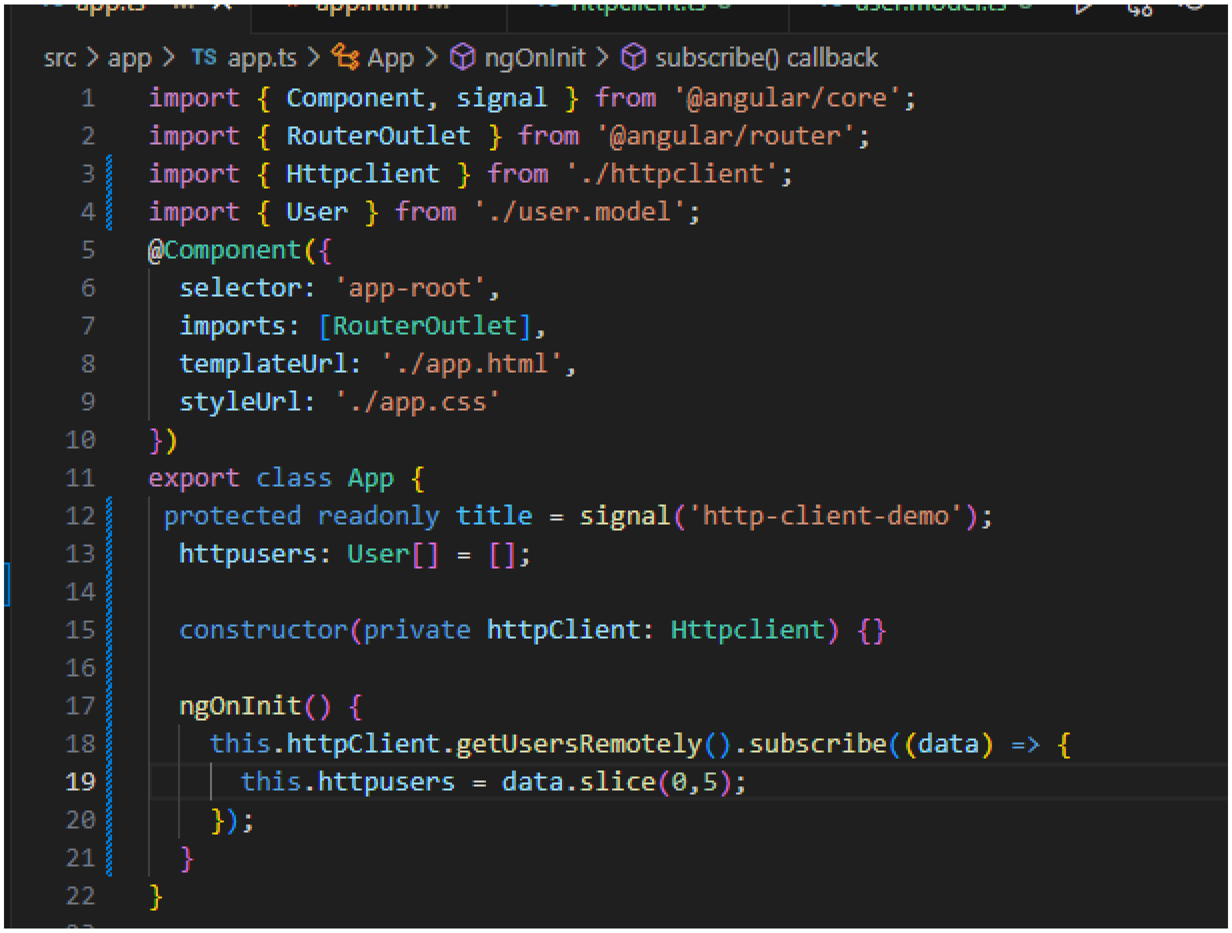


```
<h1>List from Main</h1>

<table border="1" style="width: 100%; border-collapse: collapse;">
  <thead>
    <tr>
      <th>ID</th>
      <th>Name</th>
      <th>Username</th>
      <th>Email</th>
      <th>Website</th>
    </tr>
  </thead>
  <tbody>
    @for (user of httpusers; track user.id) {
      <tr>
        <td>{{ user.id }}</td>
        <td>{{ user.name }}</td>
        <td>{{ user.username }}</td>
        <td>{{ user.email }}</td>
        <td>{{ user.website }}</td>
      </tr>
    }
  </tbody>
</table>
```

App.ts

Pereda Jan Mark S.   WD-304



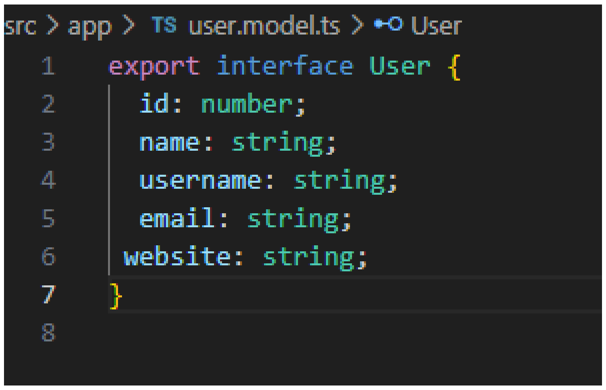
src > app > TS app.ts > App > ngOnInit > subscribe() callback

```
import { Component, signal } from '@angular/core';
import { RouterOutlet } from '@angular/router';
import { Httpclient } from './httpclient';
import { User } from './user.model';
@Component({
  selector: 'app-root',
  imports: [RouterOutlet],
  templateUrl: './app.html',
  styleUrl: './app.css'
})
export class App {
 protected readonly title = signal('http-client-demo');
  httpusers: User[] = [];

  constructor(private httpClient: Httpclient) {}

  ngOnInit() {
    this.httpClient.getUsersRemotely().subscribe((data) => {
      this.httpusers = data.slice(0,5);
    });
  }
}
```

user.model.ts

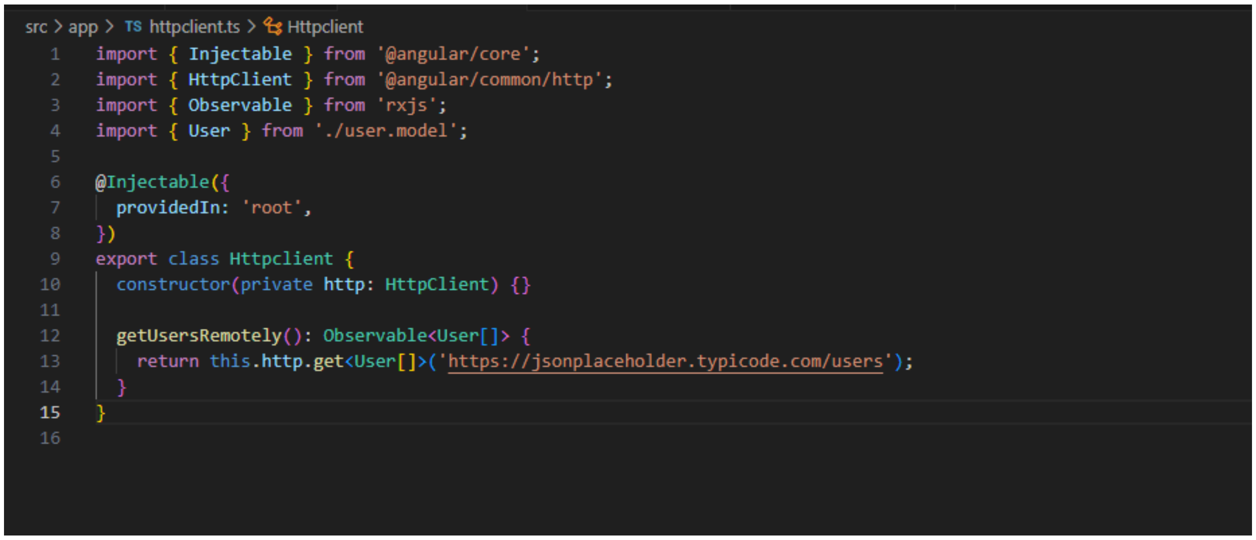


src > app > TS user.model.ts > User

```
export interface User {
  id: number;
  name: string;
  username: string;
  email: string;
 website: string;
}
```

httpclient.ts

Pereda Jan Mark S.   WD-304



src > app > TS httpclient.ts > Httpclient

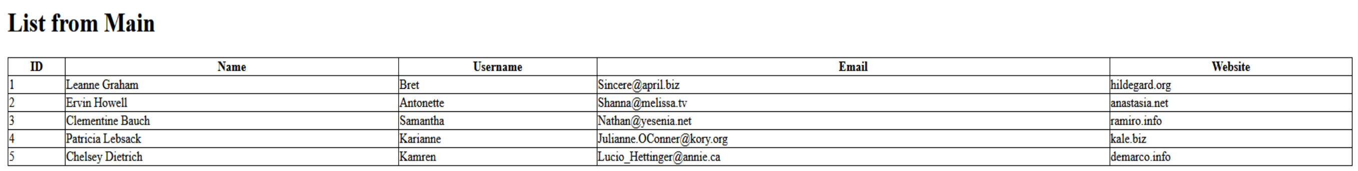
```typescript
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';
import { User } from './user.model';

@Injectable({
  providedIn: 'root',
})
export class Httpclient {
  constructor(private http: HttpClient) {}

  getUsersRemotely(): Observable<User[]> {
    return this.http.get<User[]>('https://jsonplaceholder.typicode.com/users');
  }
}

```

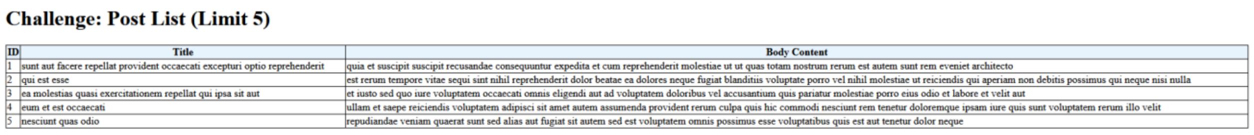
Final Output
act 1



List from Main

ID	Name	Username	Email	Website
1	Leanne Graham	Bret	Sincere@april.biz	hildegard.org
2	Ervin Howell	Antonette	Shanna@melissa.tv	anastasia.net
3	Clementine Bauch	Samantha	Nathan@yesenia.net	ramiro.info
4	Patricia Lebsack	Karianne	Julianne.OConner@kory.org	kale.biz
5	Chelsey Dietrich	Kamren	Lucio_Hettinger@annie.ca	demarco.info

Act 2



Challenge: Post List (Limit 5)

ID	Title	Body Content
1	sunt aut facere repellat provident occaecati excepturi optio reprehenderit	quia et suscipit suscipit recusandae consequuntur expedita et cum reprehenderit molestiae ut ut quas totam nostrum rerum est autem sunt rem eveniet architecto
2	qui est esse	est rerum tempore vitae sequi sint nihil reprehenderit dolor beatae ea dolores neque fugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis qui aperiam non debitis possimus qui neque nisi nulla
3	ea molestias quasi exercitationem repellat qui ipsa sit aut	et iusto sed quo iure voluptatem occaecati omnis eligendi aut ad voluptatem doloribus vel accusantium quis pariatur molestiae porro eius odio et labore et velit aut
4	eum et est occaecati	ullam et saepe reiciendis voluptatem adipisci sit amet autem assumenda provident rerum culpa quis hic commodi nesciunt rem tenetur doloremque ipsam iure quis sunt voluptatem rerum illo velit
5	nesciunt quas odio	repudiandae veniam quaerat sunt sed alias aut fugiat sit autem sed est voluptatem omnis possimus esse voluptatibus quis est aut tenetur dolor neque

Github:
https://github.com/Synchro1099/HTTP-Client-Programming?tab=readme-ov-file

Pereda Jan Mark S. WD-304

How did you use Angular’s HttpClient service to fetch data from the JSONPlaceholder API, and what HTTP method was used in the activity?

To fetch the data, I injected the **HttpClient** into my service and used the **GET** method to request the user information from the JSONPlaceholder URL.

What was the role of the service, model, and component in displaying the fetched JSON data on the web page?

Each part played a specific role in bringing the data to the screen:

- **Model**: It defined the specific structure of the user data so the application knew which fields to expect, like "id" and "name".
- **Service**: It acted as the messenger that communicated with the remote server to fetch the raw JSON data.
- **Component**: It managed the data by calling the service, saving the results, and displaying them in an organized table on the webpage.