Client Side Technologies Homework Project

**Liubov Rukhlina, LZ7EAM**

Game of Thrones Encyclopedia

# Introduction of the application and description of main functionality

Game of Thrones encyclopedia is built for the fans of a popular book and tv-series “Game of Thrones”. The author of the series – George R.R. Martin – has created a whole universe with over than 4000 characters, belonging to different houses and lands. The Game of Thrones encyclopedia allows a user to find their way around this universe.

# User Stories

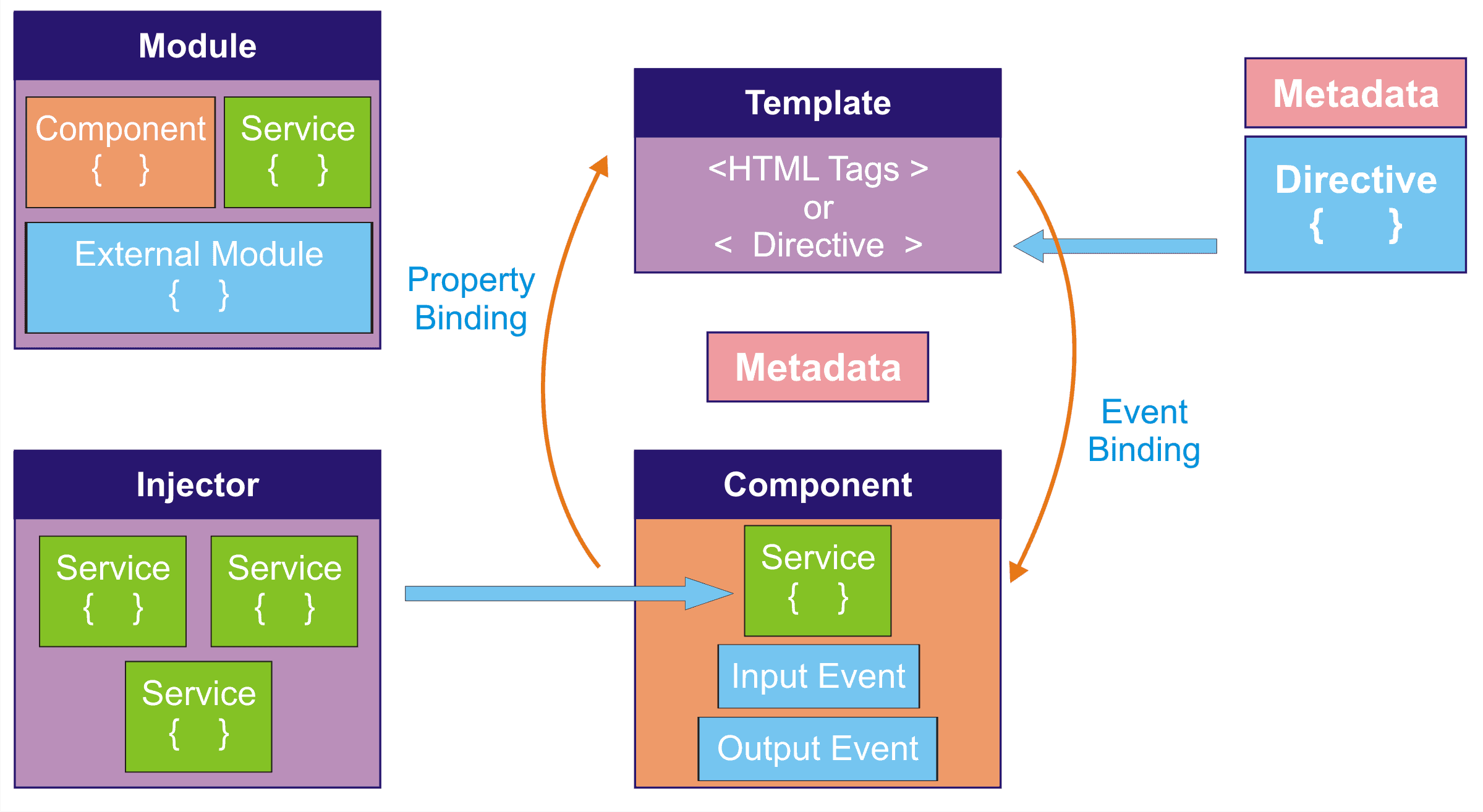
* Application is started with the display of books. From there user is able to switch between books/characters/houses views
* In those views the application displays lists of all books, characters and houses.
* The application allows to traverse the world of the book series in an interactive and ergonomic way.
* Every element of the list is clickable and allows to view the full details of the entity.
* From the details page, the user is able to navigate with one click to any related entity (e.g. to the books where the character was present or to the house they belong to).
* Moreover, search functionality was implemented to quickly access the desired character or house.

The application is built on the Ice And Fire API (https://anapioficeandfire.com/).

Description of architecture (figure!) and the introduction of main components

Angular was chosen as a technology for the project development as TypeScript is a language growing popularity on the market now and the author decided it would be valuable to include angular project to her portfolio.

For the angular project usually, it is easier to follow the standardized project architecture as the structure is modular and keeping it organized with logic/interface separation allows to develop and maintain project more efficiently. The architecture of this project is not an exception and it is illustrated in the diagram below:

[[1]](#footnote-1)

Component classes represent the views of the application and manage HTML templates. Every component includes interconnected HTML, CSS and TS files along with specifications. Main components of Game of Thrones encyclopedia are:

1. Characters: an ordered list of all game of thrones characters (just their names), paginated
2. Houses: an ordered list of all game of thrones houses (just their names), paginated
3. Books: a list of all game of thrones books (just their names)
4. Character-details: detailed view of a specific character with its full information
5. House-details: detailed view of a specific house with its full information
6. Book-details: detailed view of a specific book with its full information
7. Character-search: searching for a specific character in a list
8. House-search: searching for a specific house in a list
9. App-component: basic structure that is present in all of the views: application title and navigation bar

A list of classes created by you with a brief description of their tasks

Every component has a class with the same name of it (e.g CharactersComponent class) that implement OnInit. Their functionality was described above. Plus for every element there is a logic proving class – called service, they are: CharacterService class, BookService class and HouseService class.

Description of client server communication

In order to communicate with a remote server, HTTPClient was connected (imported). It is initialized in the constructor of every component. Url of the specific page is efined in the services, which provide logic to the application. The format of the url include: /base/collection name (e.g. /api/characters).

Observer design pattern is used to automatically update information on every change (part of RxJS).

http.get request is sent to retrieve the data from server. Response is an untyped JSON object, so special types for character, book and house were created to specify it.

To get specific element by id the Subscribe()method was used that connects the observer to observable events.

Diagram

Description automatically generated

Select and describe the whole process how a specific API call is sent to the server and how the results are received and displayed! Include source code if necessary.

Let us take a closer look on api call on a sepicific character retrieval example:

1. In order to communicate with a remote server, HTTPClient was connected (imported).

import { HttpClient, HttpHeaders } from '@angular/common/http';

in service logic files

1. It is initialized in the constructor of every component.

constructor(  
 private http: HttpClient,  
  
) {}

1. Observer design pattern is used to automatically update information on every change (part of RxJS).

import { Observable, of } from 'rxjs';

getCharacters(pageNumber: number): Observable<Character[]> {  
 const url = `${this.charactersUrl}?page=${pageNumber || 1}&pageSize=${  
 this.pageSize  
 }`;

http.get request is sent to retrieve the data from server. Response is an untyped JSON object, so special types for character, book and house were created to specify it.

return this.http.get<Character[]>(url).pipe(  
 tap((\_) => this.log('fetched characters')),  
 map((characters) => {  
 const chractersWithId = characters.map((character, index) => {  
 const id = character.url.split('/').pop() || `${index}`;  
 return { ...character, id };  
 });  
  
 return chractersWithId;  
 }),  
 catchError(this.handleError<Character[]>('getCharacters', []))  
);

1. To get specific element by id the Subscribe()method was used that connects the observer to observable events.

In components

getCharacters(): void {  
 this.characterService  
 .getCharacters(this.currentPage)  
 .subscribe((characters) => (this.characters = characters));  
}

**Online address(es) of the repository:**

<https://github.com/LiubovRukhlina/got>

**Supervisor at the department:** Mohammad Saleem

Budapest, 10 May 2022

1. https://coderlipi.com/angular/angular-architecture [↑](#footnote-ref-1)