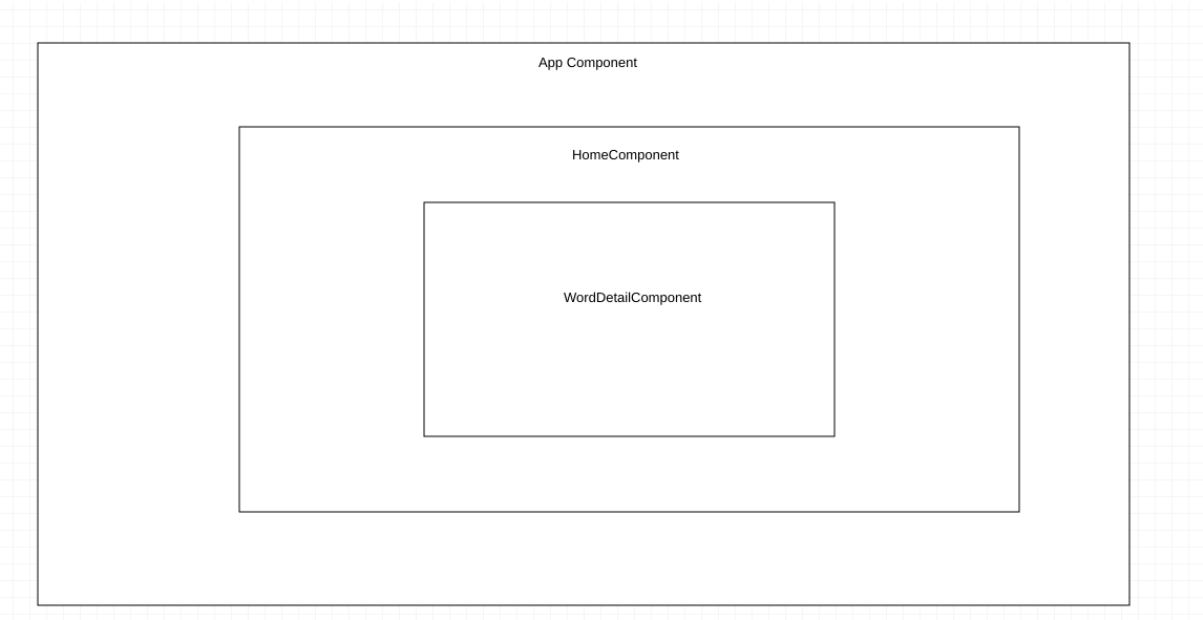
Dictionary Application

Introduction

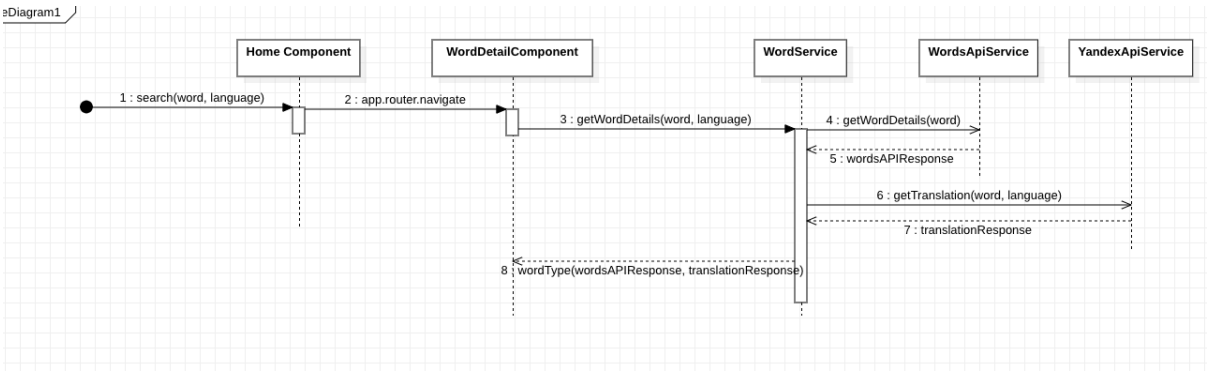
The Dictionary Application is a web-based application developed using Angular, a popular frontend framework, along with TypeScript. The primary functionality of this application is to provide translations, synonyms, antonyms, and sample phrases for a given word. The user can select both the target and source languages from a list of supported language pairs. This application makes use of two APIs: the Words API and the Yandex Dictionary API. The application is also equipped with a simple user interface to allow easy interaction with these functionalities.

Architecture

The architecture of the Dictionary Application is centered around Angular's component-based architecture. The application consists of several components, services, and models.



Application components



Searching word details

Main Components:

- AppComponent: The root component that houses all other components.
- HomeComponent: This component displays the main interface for selecting languages and entering the word.
- WordDetailComponent: This component is responsible for displaying the details of a word after it's searched.

Main Services:

- WordsApiService: This service is responsible for making HTTP requests to the Words API.
- YandexApiService: This service makes HTTP requests to the Yandex Dictionary API.
- WordService: This service combines responses from both APIs and formats them.
- AppStateService: This service maintains the state of the application.

Main Models:

These models define the structure of the responses from both APIs and how the combined data should be structured.

- TranslationResponse: Structure of response from Yandex Dictionary API.
- WordType and WordDetailsType: Structure for combined word details.
- WordsAPIResponse: Structure of response from Words API.

Classes

- **AppComponent:** The root component of the application, housing all other components.
- **HomeComponent:** Handles the language selection, word input, and triggers the search functionality.
- **WordDetailComponent:** Displays detailed information about the searched word.
- **WordsApiService:** Handles HTTP requests to the Words API.
- **YandexApiService:** Manages HTTP requests to the Yandex Dictionary API.
- **WordService:** Combines data from both APIs and formats them for display.
- **AppStateService:** Maintains the state of the application, including whether a search has been initiated.

Client-Server Communication

Communication with the server is managed through the WordsApiService and YandexApiService. These services send HTTP GET requests to their respective APIs and handle errors. The WordService then combines the responses from both services.

For instance, in YandexApiService, an HTTP GET request is made like this:

```
this.http.get<TranslationResponse>(url).pipe(
  catchError(this.handleError<TranslationResponse>('getTranslation', undefined))
);
```

Specific API Call

Consider the situation when a user enters a word for translation. The process starts from the HomeComponent where the user inputs a word and selects languages. After clicking the search button, the `search()` function triggers a router navigation to the WordDetailComponent and passes along the word and language pair as query parameters.

In WordDetailComponent's `ngOnInit()`, the `getWordDetails()` method from the WordService is called. Within this method, two API calls are made concurrently using `forkJoin()`: one to the WordsAPI and one to the Yandex Dictionary API. The responses from both APIs are then combined into a single `WordType` object and returned as an Observable.

When the Observable emits the `WordType` object, it is captured in the WordDetailComponent and displayed. Here is a relevant code example from WordDetailComponent:

```
this.route.queryParams.pipe(
  switchMap(params => {
    this.word = params['word'];
    this.languagePair = params['languagePair'];
    return this.wordService.getWordDetails(this.word, this.languagePair);
  })
).subscribe(response => {
  this.wordDetails = response;
});
```

This way, the application makes sure that both the translation and the details about the word are received and displayed at the same time.