Angular Dictionary App

Menna Elmeligy
Neptun code : AJ8AND

Introduction

The dictionary application is a user-friendly tool that provides translation, synonyms, antonyms, and sample phrases for various words. It utilizes two APIs: the Words API and the Yandex Dictionary API. The main functionality of the application includes allowing the user to select both source and target languages, fetching translation data from the API, and displaying the results to the user. In addition, the application enables users to explore synonyms, antonyms, and sample phrases associated with a given word.

Architecture

The architecture of the dictionary application follows a client-server model. The client is responsible for handling user interactions and making API requests, while the server manages communication with the external APIs and processes the retrieved data. The client-server architecture ensures separation of concerns and scalability.

The main components of the dictionary application include:

User Interface (UI): This component provides the interface for users to interact with the application. It allows them to input words, select source and target languages, and view the results.

API Manager: The API Manager component is responsible for making HTTP requests to the Words API and the Yandex Dictionary API. It handles the authentication, constructs the request URLs, and parses the responses.

Data Processor: The Data Processor component processes the data retrieved from the APIs. It extracts the necessary information such as translations, synonyms, antonyms, and sample phrases, and prepares them for display.

Result Renderer: The Result Renderer component takes the processed data and generates a visually appealing representation for the user interface. It formats the data and presents it in a readable and understandable manner.

Description:

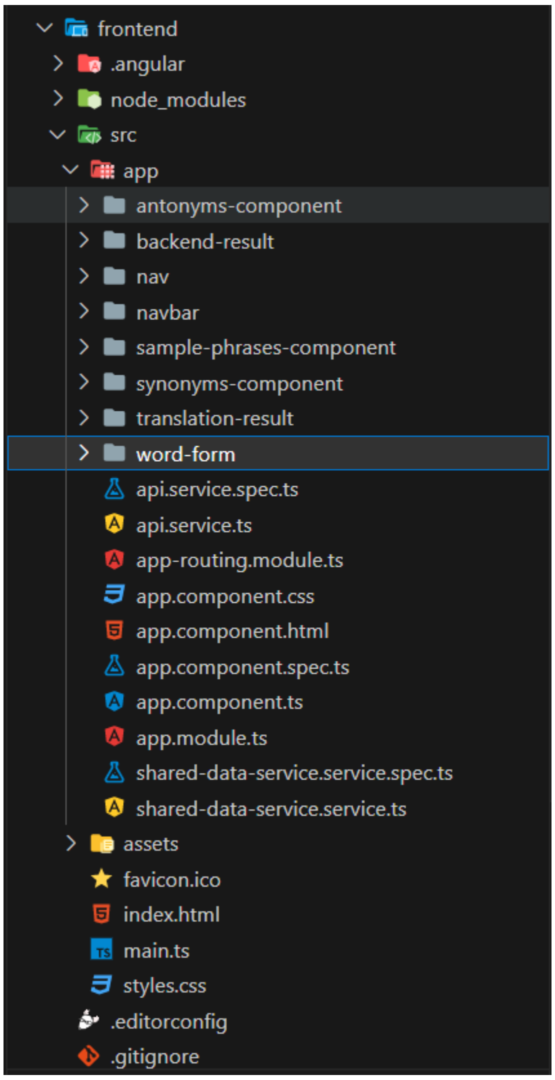
The app contains 8 components and 2 services

1-Word Form Component (word-form)

This component is responsible for displaying an input form and a submit button. When the submit button is clicked, it triggers a call to the API service (api.service.ts) to retrieve data from the API. The response from the API is stored in a variable called response.

2-API Service (api.service.ts)

This service is used by the Word Form Component to communicate with the API. It contains methods to make HTTP requests and retrieve data from the API. When the Word Form Component calls the API service, it fetches the data from the API based on the input provided by the user in the form. The retrieved data is then returned as the API response.



```
frontend
  .angular
  node_modules
  src
    app
      antonyms-component
      backend-result
      nav
      navbar
      sample-phrases-component
      synonyms-component
      translation-result
      word-form
      api.service.spec.ts
      api.service.ts
      app-routing.module.ts
      app.component.css
      app.component.html
      app.component.spec.ts
      app.component.ts
      app.module.ts
      shared-data-service.service.spec.ts
      shared-data-service.service.ts
    assets
    favicon.ico
    index.html
    main.ts
    styles.css
  .editorconfig
  .gitignore
```

3-Sample Phrases Component (sample-phrases-component)

This component receives the response from the API service and is responsible for displaying the sample phrases obtained from the response. It takes the relevant data from the response and formats it appropriately to be displayed to the user.

4-Synonyms Component (synonyms-component)

Similarly to the Sample Phrases Component, the Synonyms Component receives the response from the API service and displays the synonyms obtained from the response. It extracts the necessary data from the response and presents it in a suitable format.

5-Translation Result Component (translation-result)

This component is responsible for receiving the response from the API service and displaying the translation result. It takes the relevant data from the response and formats it for the user to view the translated output.

6-Shared Data Service (shared-data-service)

The Shared Data Service is a service that facilitates the sharing of variables between components. It allows the Word Form Component to set variables that are needed for making API calls, and the API service can retrieve those variables to initiate the API request successfully. This service acts as a mediator between the Word Form Component and the API service, enabling smooth communication and data exchange.

7-Nav Component (nav)

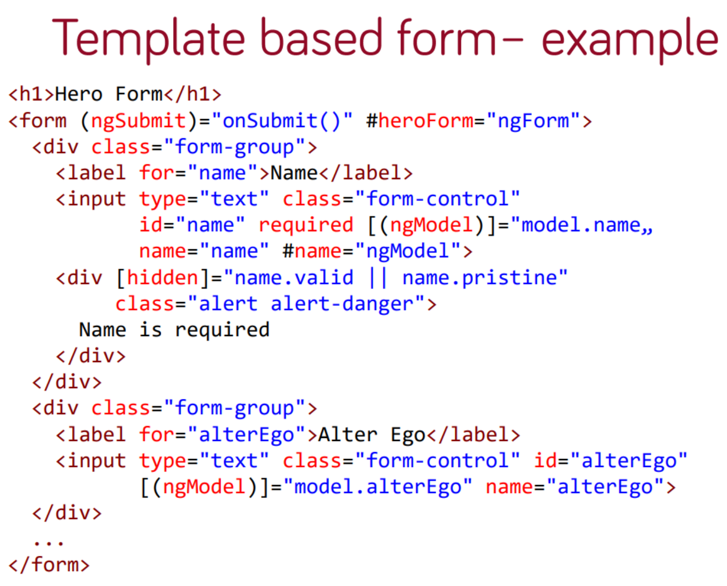
The Nav Component represents a navigation bar and contains two options: the Dictionary Page and the Translation Page. It provides users with the ability to switch between these two pages based on their needs.

8-Navbar Component (navbar)

The Navbar Component is a part of the Nav Component and serves as a visual representation of the navigation bar. It displays the available options (Dictionary Page and Translation Page) and allows users to select and navigate to the desired page.

Client-side technology related topics

as I am using Angular , using Reactive Form : in the dictionary form I am using a Form and the onSubmit to fire a function of getting the API call gone, using ngModel for data binding so if one of these properties changes , the other one will change automatically. and I used the input for entering the word and a button to fire the action.

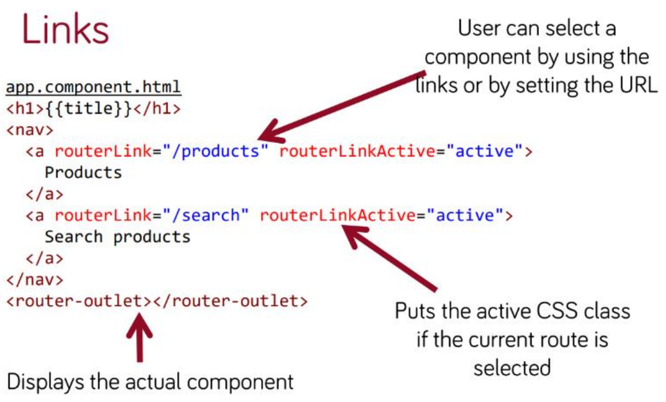


Template based form– example

```
<h1>Hero Form</h1>
<form (ngSubmit)="onSubmit()" #heroForm="ngForm">
  <div class="form-group">
    <label for="name">Name</label>
    <input type="text" class="form-control"
           id="name" required [(ngModel)]="model.name„
           name="name" #name="ngModel">
    <div [hidden]="name.valid || name.pristine"
         class="alert alert-danger">
      Name is required
    </div>
  </div>
  <div class="form-group">
    <label for="alterEgo">Alter Ego</label>
    <input type="text" class="form-control" id="alterEgo"
           [(ngModel)]="model.alterEgo" name="alterEgo">
  </div>
  ...
</form>
```

By utilizing Reactive Forms, data binding with ngModel, and form submission with a button click, you have implemented a dictionary form in Angular that captures user input, updates component properties, and triggers actions such as form submission or API calls. This approach provides a seamless user .experience and ensures data integrity and validation in your application.

utilizing Angular's routerLink directive to enable smooth navigation between pages in your application. The Nav Component contains links defined with routerLink, allowing users to click and seamlessly navigate to different pages. The Navbar Component visually displays the navigation bar across all pages, providing consistent navigation. Routing configuration maps routes to their corresponding components, determining which component to load for each page. The router initializes on application start, matching the current URL to the defined routes. Page transitions are smooth, with the content dynamically replaced and the fixed Navbar Component ensuring consistent navigation. This setup enables users to effortlessly switch between the Dictionary Page and the Translation Page, enhancing the overall user experience.


Links

app.component.html
```
<h1>{{title}}</h1>
<nav>
  <a routerLink="/products" routerLinkActive="active">
    Products
  </a>
  <a routerLink="/search" routerLinkActive="active">
    Search products
  </a>
</nav>
<router-outlet></router-outlet>
```
User can select a component by using the links or by setting the URL
Puts the active CSS class if the current route is selected
Displays the actual component

API call process:

The process of sending a specific API call to the server and receiving/displaying the results involves the following steps:

In the WordFormComponent, when the form is submitted (triggered by the onSubmit() method), the following actions occur:

The sharedDataService is used to set the source language, target language, and word entered in the form

The apiService is used to make the API call to retrieve translations.

Inside the ApiService, the getTranslations() method is called to initiate the API request:

The sharedDataService is used to get the word entered in the form.

The API endpoint URL is constructed using the word obtained from sharedDataService.

The constructed URL is then passed to the http.get() method from Angular's HttpClient module in the ApiService. This method sends the HTTP GET request to the specified API endpoint.

The server receives the API request and processes it. It retrieves the translations or other relevant data based on the provided word.

The server sends the response back to the client in JSON format.

Back in the WordFormComponent, the subscribe() method is used to listen for the response from the API call:

If the API call is successful, the response is logged to the console, and the definitions (or other relevant data) are extracted from the response and stored in the definitions variable.

If an error occurs during the API call, the error is logged to the console.

The component's template file (word-form.component.html) is bound to the definitions variable. This allows the retrieved definitions to be displayed in the UI, such as using *ngFor to iterate over the definitions array and show each definition.

Overall, this process involves using the sharedDataService to share data between components (WordFormComponent and ApiService), constructing the API URL based on the input data, making the API call using HttpClient, processing the response, and displaying the results in the component's template.

Note: The source code provided demonstrates a simplified example of the process, and it assumes the API endpoint returns the expected data structure.