Frontend Code Documentation

Source code documentation of NBP Crasher application focusing on frontend implementation and usages

Table of contents

Project Information

- **Project Name:** NBP Crasher
- **Project Manager:** Przemysław Kowalski
- **Report Date:** 17.01.2025
- **Project Duration:** 08.12.2024 – 31.01.2025

Frontend Purpose

The goal of the frontend for the NBP Crasher project is to provide a GUI (Graphical User Interface – Presentation Layer) which displays data in user-friendly interface & communicates with our API for obtaining statistical measures based on currency exchange rates retrieved from an external API (NBP API).

Used Technologies

This part of system was developed using easily available technologies.

The language used is JavaScript – standard ECMAScript 2024

UI was developed with:

- framework Angular – version 17.3.0,
- TypeScript – version 5.4.2,
- Bootstrap – version 5.3.3,
- DevExtreme Angular components – version 23.1.13.

Tests written with Karma test runner (version 5.1.0) & Jasmine testing framework (version 5.1.0).

For final deployment we used contenerization using Docker

Code documentation

Application Configuration and Constants

appConfig: Application configuration with providers.

appName: Application name.

defaultCurrencyFrom: Default "from" currency.

defaultCurrencyTo: Default "to" currency.

minDateFrom: Minimum date selectable (Based on NBP API docs).

timeFrames: Array of available time frames.

defaultTimeFrameIndex: Default time frame index (1W).

defaultTimeFrame: Default time frame (1W).

dateFormat: Date format.

blankData: Placeholder for blank data.

Models and Types

SupportedCurrency: Supported currencies type definition.

Currency: Currency type with name property.

supportedCurrencies: Array of supported currencies.

TimeFrame: Time frame type extending DxTabsItem with optional dateFrom and dateTo properties.

Statistics: Statistics type definition.

Session: Session type definition.

ChangeDistribution: Change distribution type definition.

Data: Data type definition, aggregating statistics, sessions, and changes distribution.

BaseService

apiErrors: Base API errors.

notificationOverride: Override settings for notifications.

defaultError: Default error configuration.

generateParams(filters?: unknown): HttpParams Generates HttpParams from given filters. *Parameters:*

- `filters` (optional): Filters to be converted into HttpParams.

Returns:

- `HttpParams`: The generated HttpParams.

catchCustomError(error: HttpErrorResponse): Observable<void> Handles custom HTTP errors and displays appropriate notifications. *Parameters:*

- `error`: The HTTP error response to handle.

Returns:

- `Observable<void>`: An observable that throws the processed error message.

StatsService

baseRoutePath: Base route path for the stats API.

get(filters: GetStatsRequest): Promise<Data> Fetches statistics data based on given filters. *Parameters:*

- `filters`: Filters to apply to the stats request.

Returns:

- `Promise<Data>`: A promise that resolves to the fetched data.