Backend Code Documentation

Source code documentation of NBP Crasher application focusing on backend functions, implementation and usages

Spis treści

[Project Information 3](#__RefHeading___Toc619_1849182835)

[Backend Purpose 3](#__RefHeading___Toc621_1849182835)

[Used Technologies 3](#__RefHeading___Toc623_1849182835)

[Code documentation 4](#__RefHeading___Toc625_1849182835)

[API 4](#__RefHeading___Toc627_1849182835)

[Data analysis 4](#__RefHeading___Toc629_1849182835)

[json\_to\_data\_frame 4](#__RefHeading___Toc631_1849182835)

[calculate\_statistical\_measures 4](#__RefHeading___Toc633_1849182835)

[count\_session 4](#__RefHeading___Toc635_1849182835)

[calculate\_distribution 5](#__RefHeading___Toc637_1849182835)

[create\_dynamic\_ranges 5](#__RefHeading___Toc643_1849182835)

[calculate\_statistics 5](#__RefHeading___Toc649_1849182835)

# Project Information

* **Project Name:** NBP Crasher
* **Project Manager:** Przemysław Kowalski
* **Report Date:** 07.01.2025
* **Project Duration:** 08.12.2024 – 31.01.2025

# Backend Purpose

The goal of the backend for the NBP Crasher project is to provide a functional and complete API for obtaining statistical measures based on currency exchange rates retrieved from an external API ([NBP API](https://api.nbp.pl/)).

# Used Technologies

This part of system was developed using easily available technologies.

The language used is [Python](https://www.python.org/) – version 3.12

API was developed with framework [FastAPI](https://fastapi.tiangolo.com/) in version 0.115.6

Data analysis was perfomed with help of following external packages:

* [numpy](https://numpy.org/) – version 2.2.1
* [pandas](https://pandas.pydata.org/) – version 2.2.3

Web server hosting our API implementation is [uvicorn](https://www.uvicorn.org/) – version 0.34.0

For final deployment we used contenerization using [Docker](https://www.docker.com/)

# Code documentation

## API

API documentation is put in other file in this file directory   
 under name „NBP Crasher.pdf”

## Data analysis

### **json\_to\_data\_frame**

Converts a JSON object into a pandas DataFrame.

Parameters:   
 json\_content (dict): Input JSON data   
 Returns:   
 pd.DataFrame: DataFrame containing the JSON data

### **calculate\_statistical\_measures**

Calculates basic statistical measures for numerical data.

Parameters:   
 data (set): Input numerical values set   
 Returns: Dictionary containing:   
 `mode`: Dictionary of mode values and frequencies   
 `standard\_deviation`: Standard deviation (4 decimal places)   
 `variation\_coefficient`: Variation coefficient as percentage   
 (4 decimal places)   
 `median`: Median value (4 decimal places)   
 Raises:   
 ValueError: If data is None

### **count\_session**

Counts increasing, decreasing and unchanged sessions.

Parameters:   
 data (pd.Series): Input numerical values series   
 Returns: Dictionary containing:   
 `increasing\_sessions`: Count of value increases   
 `decreasing\_sessions`: Count of value decreases   
 `no\_change\_sessions`: Count of unchanged values   
 Raises:   
 ValueError: If data is None

### **calculate\_distribution**

Calculates distribution of absolute currency rate changes.

Parameters:

currency\_rate (pd.Series): Currency rate values

Returns: List of dictionaries containing:  
 `rangeBegin`: Range interval start   
 `rangeEnd`: Range interval end  
 `value`: Count in range   
 Raises:   
 ValueError: If currency\_rate is None

### **create\_dynamic\_ranges**

Creates dynamic range boundaries and labels.

Parameters:  
 data (pd.Series): Input data   
 n\_ranges (int, optional): Number of ranges (default: 14)   
 Returns: Tuple containing:   
 boundaries: Range boundary values list   
 labels: Formatted range labels list   
 Raises:  
 ValueError: If data is None

### **calculate\_statistics**

Calculates comprehensive statistics for one or two currencies.

Parameters:  
 first\_currency (dict): First currency data   
 second\_currency (dict, optional): Second currency data   
 Returns: Dictionary containing:   
 `statistics`: Statistical measures   
 `sessions`: Session counts   
 `changes\_distribution`: Rate changes distribution   
 Raises:   
 ValueError: If first\_currency is None   
 KeyError: If data structure invalid