A picture containing sky, outdoor

Description automatically generated

[1. Вступ 2](#_Toc88105352)

[2. Вхідні дані 3](#_Toc88105353)

[3. Задача 4](#_Toc88105354)

[4. Технічний опис задач 5](#_Toc88105355)

[*4.1.* *Розрахунок бенчмарка за заданим часовим проміжком* 5](#_Toc88105356)

[*4.2.* *Розрахунок агрегованих даних бенчмарку в обраному часовому інтервалі* 8](#_Toc88105357)

[5. Технічні вимоги 12](#_Toc88105358)

[6. Критерії оцінювання 13](#_Toc88105359)

# Вступ

Вітаємо з успішним проходженням першого туру!

Давайте продовжимо нашу подорож у світ фінансів, портфелів та цін. Сьогодні наші розрахунки будуть трохи більш складнішими та масштабними, проте ми впевнені, що ви зможете впоратися з ними!

Сьогоднішнє завдання буде логічним продовженням того, що ми робили у 1 турі з розрахунком середньої ціні.

# Вхідні дані

Вхідними даними для розрахунків є дані про ціни на фінансові інструменти. Кожна ціна включає в себе наступні ключові дані:

* Значення ціни — ***price value*** *(pv)*;
* Відмітка моменту отримання ціни — ***price datetime*** *(pdt).*

Файл вхідних даних не змінився та знаходиться у *SC.DevChallenge.Api/Input/data.csv*

# Задача

Завдання складається з двох частин, які необхідно виконати. Результатом кожного з завдань буде endpoint веб-сервісу, який можна використовувати для розрахунків:

* **Розрахунок контрольного показника (далі за текстом бенчмарка).**

Очікуваний результат:

|  |  |  |
| --- | --- | --- |
| ***GET*** | /api/prices/benchmark?portfolio={portfolio}&date={datetime} | Повертає бенчмарк за заданим Портфелем та таймслотом, який відповідає часу вказаному у запиті. |

* **Розрахунок агрегованого бенчмарка.**

Очікуваний результат:

|  |  |  |
| --- | --- | --- |
| ***GET*** | /api/prices/aggregate?portfolio={portfolio}&startdate={datetime\_start}&enddate= {datetime\_end}&intervals={intervals} | Повертає обрану кількість агрегованих бенчмарків, що відповідають портфелю в обраний інтервал часу. |

Детальний опис і приклади того, як повинні виконуватися розрахунки, можна знайти нижче.

# Технічний опис задач

## *Розрахунок бенчмарка за заданим часовим проміжком*

Розрахунок часових проміжкуів залишився незмінним з попереднього туру. Нашим першим завданням буде розрахунок бенчмарку для кожного часового проміжку. Результат - це середнє зі знайдених цін за заданим портфелем у часовому проміжку.

|  |  |  |
| --- | --- | --- |
| ***GET*** | /api/prices/benchmark?portfolio={portfolio}&date={datetime} | Повертає бенчмарк за заданим Портфелем та таймслотом, який відповідає часу вказаному у запиті. |

***Формат запиту:***

|  |  |  |
| --- | --- | --- |
| ***Назва*** | ***Тип*** | ***Опис*** |
| Portfolio | String | Портфоліо цінних паперів |
| Date | DateTime | Дата для підрахунку |

***Приклади запитів:***

|  |  |
| --- | --- |
| ***Звичайний*** | api/prices/benchmark?portfolio=Fannie Mae&date=15/03/2018 17:34:50 |
| ***Закодований*** | api/prices/benchmark?portfolio=Fannie+Mae&date=15%2F03%2F2018+17%3A34%3A50 |

**Результат**:

* **Запит було успішно оброблено**

**Status Code**: 200 (OK)

|  |  |  |
| --- | --- | --- |
| ***Назва*** | ***Формат*** | ***Опис*** |
| Date | "dd/MM/yyyy HH:mm:ss" | Дата початку відповідного часового інтервалу |
| Price | “xx.yy” | Бенчмарк на інтервалі, округлений до 2-х знаків після коми. |

* **Приклад успішної відповіді на запит (JSON):**

{

"date": "15/03/2018 17:26:40",

"price": 133.71

},

де "15/03/2018 17:26:40" - це стартова точка часового проміжку #637, а 133.71 - це середнє значення цін за портфелем ‘Fannie Mae’ на часовому проміжку #637, які знаходяться між верхньою та нижньою межою обчислення.

* **Помилка у обробці запиту через нестачу даних**

**Status Code:** 404 (Not found).

Щоб розрахувати бенчмарк правильно, спершу необхідно позбутися всіх крайніх (віддалених) значень цін. Це потрібно для того, щоб отримати більш високу точність розрахунку, оскільки крайні значення можуть вносити сильні флуктуації у фінальне значення бенчмарку.

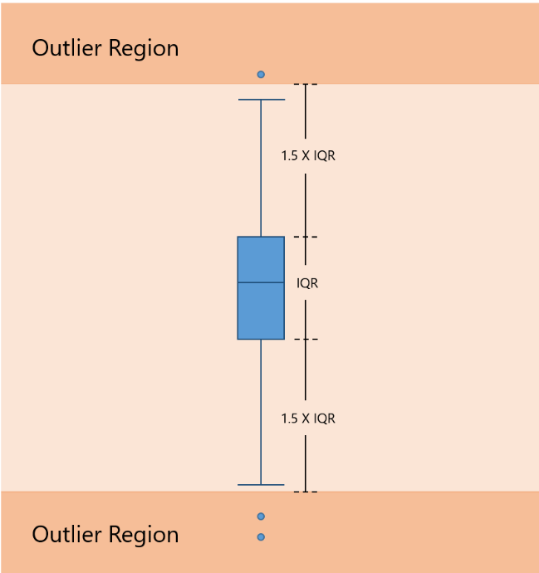
Крайні значення усуваються для кожного часового проміжку незалежно від інших часових проміжків, за таким алгоритмом:

* **Розрахувати перший та третій квартилі (Q1 та Q3) залежно від ціни pavу зростаючому порядку.**

Інтерквартильний розмах(IQR) визначається як IQR = pav[Q3] – pav[Q1].

Нижня межа визначається як pav[Q1] - *1.5\**IQR, верхня межа як pav[Q3]+*1.5\**IQR.

Принцип показаний на картинці нижче.



* **Усі значення цін у часовому проміжку, які менші за pav[Q1] - *1.5\**IQRабо більші за pav[Q3]+*1.5\**IQR, виключаються з розрахунків бенчмарку;**
* **Бенчмарком вважається середнє значення від усіх цін, що залишилися.**

## *Розрахунок агрегованих даних бенчмарку в обраному часовому інтервалі*

Наступне завдання, яке ми виконаємо, це можливість отримувати бенчмарк в певному часовому проміжку, розбитому на необхідну нам кількість інтервалів.

|  |  |  |
| --- | --- | --- |
| ***GET*** | /api/prices/aggregate?portfolio={portfolio}&startdate= {datetime\_start}&enddate= {datetime\_end}&intervals={intervals} | Повертає середню ціну, що відповідає обраній відмітці DateTime. Вхідні параметри не чутливі до регістру. |

***Формат запиту:***

|  |  |  |
| --- | --- | --- |
| ***Назва*** | ***Тип*** | ***Опис*** |
| Portfolio | String | Портфоліо цінних паперів |
| StartDate | DateTime | Дата початкової точки |
| EndDate | DateTime | Дата кінцевої точки |
| Intervals | Int32 | Кількість інтервалів для розбивки |

***Приклади запитів:***

|  |  |
| --- | --- |
| ***Звичайний*** | api/prices/aggregate?portfolio=Fannie Mae&startdate=06/10/2018 00:00:00&enddate=13/10/2018 00:00:00&intervals=7 |
| ***Закодований*** | api/prices/aggregate?portfolio=Fannie+Mae&startdate=06%2F10%2F2018+00%3A00%3A00&enddate=13%2F10%2F2018+00%3A00%3A00&intervals=7 |

**Результати**:

* **Запит було успішно оброблено**

**Status Code**: 200 (OK)

**Приклад успішної відповіді на запит (JSON):**

[

{

"date": "06/10/2018 19:40:00",

"price": 381.47

},

{

"date": "07/10/2018 20:40:00",

"price": 203.9

},

....

....

{

"date": "12/10/2018 00:40:00",

"price": 465.67

},

{

"date": "12/10/2018 22:53:20",

"price": 187.77

}

],

де "06/10/2018 19:40:00" є початком часового інтервалу #2409 першої групи [2401 - 2409],

"07/10/2018 20:40:00" є початком часового інтервалу #2410 другої групи [2410 - 2418],

...

"12/10/2018 22:53:20 є початком часового інтервалу #2452 останньої групи [2455 - 2462].

* **Помилка у обробці запиту через нестачу даних:**

**Status Code:** 404 (Not found).

Алгоритм повинен працювати за наступною логікою:

* Розрахувати перший часовий проміжок, з якого починається відлік часу *tss* = **dtToTs**(*dts*);
* Розрахувати останній часовий проміжок *tse* = **dtToTs**(*dte*);
* Отримати N часових проміжків, які необхідно розділити на *pn* приблизно рівних груп *tsg;*
* Для кожної групи *tsg* необхідно порахувати та вивести її бенчмарк та стартову точку останнього часового проміжку, який до неї входить.

A picture containing text, antenna

Description automatically generated

**Приклад*:***

Якщо ми хочемо отримати агреговані дані по бенчмарку ціни для портфеля “AXA”, часового проміжку між ***2018-01-09 00:00:01 та 2018-01-15 23:59:59,*** розбиті на 7 інтервалів (по одному на кожен день), то наш метод буде виглядати як:

***TimeslotAggr(****AXA****, 2018-01-09 00:00:01, 2018-01-15 23:59:59, 7)***

В даному випадку в інтервал між ***2018-01-09 00:00:01* та *2018-01-15 23:59:59* потрапляє 61 часовий проміжок. Далі їх необхідно розділити на 7 груп та порахувати бенчмарк для кожної окремо.**

**Якщо кількість часових проміжків не ділиться порівну на кількість груп, то групи потрібно наповнювати за залишковим принципом. Наприклад, 10/4 повинно дати групи розміром 3, 3, 2, 2.**

**Умовні позначення:**

Формат дати: "dd/MM/yyyy HH:mm:ss"

Якщо не знайдено даних, то код статусу повинен бути 404.

# Технічні вимоги

Вихідний код проєкта повинен бути побудований на базі шаблона зі збереженням структури репозиторію:

*https://github.com/SimCorp/DevChallenge*

Вихідний код готового завдання має бути завантажений як git репозиторій з відкритим доступом (наприклад, через форк вищенаведеного репозиторію, якщо ви маєте GitHub аккаунт).

Тобто, повинна бути можливість клонувати репозиторій без додаткової авторизації. Наприклад:

*git clone https://github.com/olexta/DevChallenge*

Фреймворк, який використовується - **.NET 5.**

Повинна бути можливість зібрати Docker образ API, за допомогою наступних команд, виконаних з директорії SC.DevChallenge.Api (як і в шаблонному проекті):

*dotnet publish -c Release*

*docker build -t sc-dev-challenge .*

Для перевірки розв’язку завдання контейнер з API буде запускатися наступним чином:

*docker run -it --rm -p 5000:80 sc-dev-challenge*

API endpoint завдання повинен бути доступним за таким URI:

*http://localhost:5000/api/prices/benchmark*

*http://localhost:5000/api/prices/aggregate*

# Критерії оцінювання

|  |  |
| --- | --- |
| ***Критерій*** | ***Максимальна оцінка*** |
| Правильний розв’язок Задачі 1 | 30 |
| Правильний розв’язок Задачі 2 | 30 |
| Дотримання принципів ООП у коді | 10 |
| Оптимізація алгоритмів розрахунків | 10 |
| Наявність документації в коді | 5 |
| Наявність юніт тестів | 5 |
| Наявність інтеграційних тестів (API tests) | 5 |
| Архітектура бази даних | 5 |