Zadanie

Napisz serwis http, który będzie przygotowywał raport o transakcjach klienta na podstawie otrzymanych danych wg specyfikacji poniżej. Możesz użyć dowolnego frameworku dla Pythona 3.

Request (content_type: application/json)
POST /report

```
{
pay_by_link: [
      {
            created_at: string(date-time)
            currency: string
            amount: int
            description: string
            bank: string
      }
],
dp: [
      {
            created_at: string(date-time)
            currency: string
            amount: int
            description: string
            iban: string
      }
],
card: [
      {
            created_at: string(date-time)
            currency: string
            amount: int
            description: string
            cardholder_name: string
            cardholder_surname: string
            card_number: string
      }
]
}
```

Obiekty

PayByLink

```
created_at: string(date-time) # data w formacie
2021-05-14T08:27:09Z (może być w różnych strefach)
currency: string # kod waluty z listy [“EUR”,”USD”, “GBP”,  “PLN”]
amount: int # kwota w najniższym nominale np. groszach: 1000 to
10.00 zł
```

```
description: string # opis płatności np. Order 17
bank: string # nazwa banku np. mbank
```

DirectPayment

```
created_at: string(date-time) # data w formacie
2021-05-14T08:27:09Z (może być w różnych strefach)
currency: string # kod waluty z listy [“EUR”,”USD”, “GBP”,  “PLN”]
amount: int # kwota w najniższym nominale np. groszach: 1000 to
10.00 zł
description: string # opis płatności np. Order 17
iban: string # numer konta np. DE91100000000123456789
```

Card

```
created_at: string(date-time) # data w formacie
2021-05-14T08:27:09Z (może być w różnych strefach)
currency: string # kod waluty z listy [“EUR”,”USD”, “GBP”,  “PLN”]
amount: int # kwota w najniższym nominale np. groszach: 1000 to
10.00 zł
description: string # opis płatności np. Order 17
cardholder_name: string # imię posiadacza karty np. Jan
cardholder_surname: string # nazwisko posiadacza karty np. Nowak
card_number: string # number karty 1111111111111111
```

Dane powinny być zwalidowane, w przypadku nieprawidłowych danych należy zwrócić błąd **400 Bad Request**.

Serwis ma zwracać raport o płatnościach danego klienta w porządku chronologicznym, który będzie listą z informacjami o płatnościach w jednorodnym formacie (PaymentInfo).

Response:

```
 [
     {
          date: string(date-time)
          type: string
          payment_mean: string
          description: string
          amount: int
          currency: string
          amount_in_pln: int
     }
]
```

PaymentInfo:

```
date: string(date-time) # data w formacie 2021-05-14T08:27:09Z
z pola created_at (znormalizowana do UTC)
type: string # typ płatności z listy [“pay_by_link”, ”card”, “dp”]
payment_mean: string # informacja o sposobie płatności:
   - dla PayByLink: bank
```

```
   - dla DirectPayment: iban
   - dla Card: ‘cardholder_name cardholder_surname
     masked_card_number’ np. ‘Jan Nowak 1111********1111’
description: string # opis płatności z pola description
amount: int
currency: string
amount_in_pln: int # kwota po przewalutowaniu na pln (w groszach)
zgodnie z kursem z dnia wykonania płatności
```

Do przewalutowania kwoty płatności na PLN potrzebne będą dane o kursie danej waluty z dnia płatności . Zintegruj się w tym celu z http://api.nbp.pl/. Kwoty po przewalutowaniu zaokrąglij w dół, do groszy.

Przykład:

Request:

```
{
pay_by_link: [
      {
            created_at: 2021-05-13T01:01:43-08:00
            currency: EUR
            amount: 3000
            description: Abonament na siłownię
            bank: mbank
      }
],
dp: [
      {
            created_at: 2021-05-14T08:27:09Z
            currency: USD
            amount: 599
            description: FastFood
            iban: DE91100000000123456789
      }
],
card: [
      {
            created_at: 2021-05-13T09:00:05+02:00
            currency: PLN
            amount: 2450
            description: REF123457
            cardholder_name: John
            cardholde_surname: Doe
            card_number: 2222222222222222
      },
      {
            created_at: 2021-05-14T18:32:26Z
            currency: GBP
```

```
            amount: 1000
            description: REF123456
            cardholder_name: John
            cardholde_surname: Doe
            card_number: 1111111111111111
      },
]
}
```

Response:

```
 [
      {
            date: 2021-05-13T07:00:05Z
            type: card
            payment_mean: John Doe 2222********2222
            description: REF123457
            currency: PLN
            amount: 2450
            amount_in_pln: 2450
      },
      {
            date: 2021-05-13T09:01:43Z
            type: pay_by_link
            payment_mean: mbank
            description: Abonament na siłownię
            currency: EUR
            amount: 3000
            amount_in_pln: 13494
      },
      {
            date: 2021-05-14T08:27:09Z
            type: dp
            payment_mean: DE91100000000123456789
            description: FastFood
            currency: USD
            amount: 599
            amount_in_pln: 2219
      },
      {
            date: 2021-05-14T18:32:26Z
            type: card
            payment_mean: John Doe 1111********1111
            description: REF123456
            currency: GBP
            amount: 1000
            amount_in_pln: 5208
      }
]
```

Będziemy oceniać:

- jakość kodu
- użycie odpowiednich bibliotek
- jakość testów

Zadanie dodatkowe - na szóstkę :)

Dodaj:

- nowy endpoint POST, np. **`/customer-report`**, który serwowałby funkcjonalność endpointu **`/report`** rozszerzoną o możliwość przyjmowania id klienta (dodaj pole “customer_id” do payloadu) i zapisywał ostatnio wysłany raport (np. w bazie danych lub pamięci)
- nowy endpoint GET, np. **`/customer-report/[customer_id]`**, którym będzie można pobrać ostatni wysłany raport dla danego klienta