Task

Implement http service, which will prepare a report containing client's transactions given request as in specification below. You can use any Python 3 web framework.

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
      }
]
}
```

Objects specification

PayByLink

created_at: string(date-time) # data in format 2021-05-14T08:27:09Z (can be from different time zones)
currency: string # currency code from list: [“EUR”, ”USD”, “GBP”, “PLN”]
amount: amount in the lowest denomination ex. polish **grosz**: 1000 is 10.00 zł

```
description: string # payment description ex. Order 17
bank: string # bank name ex. mbank
```

DirectPayment

```
created_at: string(date-time) # date in format
2021-05-14T08:27:09Z (can be from different time zones)
currency: string # currency code from list: [“EUR”, ”USD”, “GBP”,
“PLN”]
amount: amount in the lowest denomination ex. polish grosz: 1000
is 10.00 zł
description: string # payment description ex. Order 17
iban: string # account number ex. DE91100000000123456789
```

Card

```
created_at: string(date-time) # date in format
2021-05-14T08:27:09Z (can be from different time zones)
currency: string # currency code from list [“EUR”, ”USD”, “GBP”,
“PLN”]
amount: amount in the lowest denomination ex. polish grosz: 1000
is 10.00 zł
description: string # payment description ex. Order 17
cardholder_name: string # ex. John
cardholder_surname: string # ex. Smith
card_number: string # ex. 1111111111111111
```

Date should be validated, in case of invalid date, api should respond with error **400 Bad Request**.

Service should respond with a report of the client's payments in chronological order, as a list with payment data in uniform format (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) # date in the format 2021-05-14T08:27:09Z
from field created_at (in the UTC time zone)
```

```
type: string # payment type from list: [“pay_by_link”, ”card”,
“dp”]
payment_mean: string:
   - for PayByLink: bank
   - for DirectPayment: iban
   - for Card: ‘cardholder_name cardholder_surname
     masked_card_number’ np. ‘John Smith 1111********1111’
description: string # value of ‘description’ field
amount: int
currency: string
amount_in_pln: int # amount after currency conversion to pln (in
polish grosz) with exchange rate as of the day of payment
```

To make currency conversion to PLN, exchange rate from the day of payment will be needed. To obtain this rate, integrate with http://api.nbp.pl/. Amounts after conversion should be rounded down, to polish **grosz**.

Example:

Request:

```
{
pay_by_link: [
      {
            created_at: 2021-05-13T01:01:43-08:00
            currency: EUR
            amount: 3000
            description: Gym membership
            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: Gym membership
            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
        }
]
```

What will be rated:

- code quality
- proper libraries usage
- tests quality

Additional task (Optional):

Add:

- new POST endpoint, ex. **/customer-report**, which would serve functionality as in /report endpoint, but extended - customer_id can be passed and last report will be saved (in database or in memory)
- new GET endpoint, ex. **/customer-report/[customer_id]**, which can be used to retrieve last saved report for this customer