Building a Payment Gateway

Background

E-Commerce is experiencing exponential growth and merchants who sell their goods or services online need a way to easily collect money from their customers.

We would like to build a payment gateway, an API based application that will allow a merchant to offer a way for their shoppers to pay for their product. You can use the software language of your choice. Source code with no binary must be provided.

Requirement

The software that you will develop will allow a merchant to process payments through a REST API.

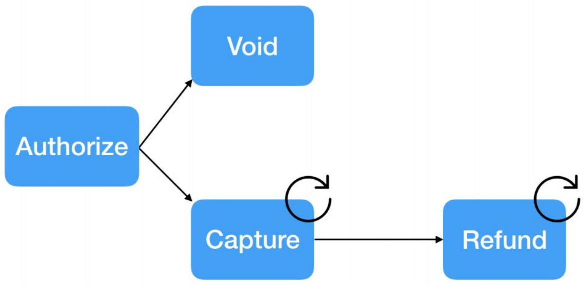
It should simulate the following payment flow:

- A merchant requests an authorisation through the /authorize call. This call contains the customer credit card data as well as an amount and currency. It will return a unique ID that will be used in all next API calls.
- The /void call will cancel the whole transaction without billing the customer. No further action is possible once a transaction is voided.
- The /capture call will capture the money on the customer bank. It can be called multiple times with an amount that should not be superior to the amount authorised in the first call. For example, a 10£ authorisation can be captured 2 times with a 4£ and 6£ order.
- The /refund call will refund the money taken from the customer bank account. It can be also called multiple times with an amount that can not be superior to the captured amount. For example, a 10£ authorisation, with a 5£ capture, can only be refunded of 5£. Once a refund has occurred, a capture cannot be made on the specific transaction.

In order to test specific edge case, your application must trigger errors based on the credit card number:

- 4000 0000 0000 0119: authorisation failure
- 4000 0000 0000 0259: capture failure
- 4000 0000 0000 3238: refund failure

Your application should handle multiple transactions at the same time, at all different transaction stages.



Endpoints

- /authorize
    - Input
        - Credit card data
        - Card number
        - Expiry month and date
        - CVV
    - Amount and currency
    - Output
        - Unique ID
        - Success or error
        - Amount and currency available

- /capture
    - Input
        - Authorization unique ID
        - Amount
    - Output
        - Success or error
        - Amount and currency available
- /void
    - Input
        - Authorization unique ID
    - Output
        - Success or error
        - Amount and currency available

- /refund
    - Input
        - Authorization unique ID
        - Amount
    - Output
    - Success or error
    - Amount and currency available

Consideration
Include whatever documentation/notes you feel is appropriate. This should include some details of assumptions made, areas you feel could be improved.

Extra mile bonus points
In addition to the above, time permitting, consider the following suggestions for taking your implementation a step further:

- Implement Luhn check on credit card number
- Client (merchant) authentication
- Application logging
- Containerization
- Anything else you feel may benefit your solution from a technical perspective