PAYMENT API

This repository contains source code and documentation for a payments RESTFull api. Allows create, read, update, get and list payments. the operations are idempotent.
Contains a docker file and a health check for production readiness. To persist data uses Postgres sql.

Dependencies

This project requires the following dependencies:

- Golang
- PostgresSQL
- Docker

In the docker-compose.yaml, there is a postgresql container configured. Install docker compose and run

```
docker-compose up
```

How to install

Run docker-compose or local postgresql.

```
make install
make build
```

It will generate a executable file called payment-api.

How to run

```
Usage: payment-api [OPTIONS]

Allows create, read, update, get and list operations. Uses postgresql and got a health check.

Options:
    --port               HTTP port for the app (env $PORT) (default 8081)
    --path-prefix        Version of the API start with a /. The endpoints created will start by it. (env $path-prefix) (default "/v1")
    --write-timeout      number of seconds the http call waits writing until it times out. (env $WRITE_TIMEOUT) (default 10)
    --read-timeout       number of seconds the http call waits reading until it times out. (env $READ_TIMEOUT) (default 10)
    --idle-timeout       number of seconds the http call waits idling until it times out. (env $IDLE_TIMEOUT) (default 10)
    --db-address         the db address with the port number - eg.  127.0.0.1:5432 (env $DB_ADDRESS) (default "127.0.0.1:5432")
    --db-username        postgresql username (env $DB_USERNAME) (default "test")
    --db-password        postgresql password (env $DB_PASSWORD) (default "example")
    --db-name            the name of the database (env $DB_NAME) (default "test")
    --log-level          Desired log level, - eg. info, warn, error (env $LOG_LEVEL) (default "debug")
    --graceful-timeout   the duration for which the server gracefully wait for existing connections to finish - e.g. 15s or 1m (env $GRACEFUL_TIMEOUT) (default 10
```

Run the app with or without options.

```
./payments-api
```

RESTFull API

For a more detailed endpoint information, use swagger and import the payment.swagger.yaml.

Architecture

1 . App

This app was build using Golang with the following dependecies. - mux - HTTP router used in this project. Another solution was to use grpc-gateway to build my REST endpoint but was discard since would create additional GRPC endpoints. - go-pg - PostgresSQL driver for golang. Made development of the app much quicker since it has mapping between Query/Result to go struct.

2. Database

The database used in this project is PostgresSQL. It was the choose solution because is a SQL DB, has a good performance, is scalable and guarantees ACID operations.

Routes

GET Methods

- /v1/payments?limit=100&offset=0

Returns all the payments. Query params are optional. The default limit is 100 and max is 10000. Offset default value is 0.

- /v1/payment/{paymentID}

Returns one payment

POST Methods

- /v1/payment

Creates a new payment. A example payload would be like the following:

```
{
      "type": "Payment",
      "id": "4ee3a8d8-ca7b-4290-a52c-dd5b6165ec43",
      "version": 0,
      "organisation_id": "743d5b63-8e6f-432e-a8fa-c5d8d2ee5fcb",
      "attributes": {
        "amount": "100.21",
        "beneficiary_party": {
          "account_name": "W Owens",
          "account_number": "31926819",
          "account_number_code": "BBAN",
          "account_type": 0,
          "address": "1 The Beneficiary Localtown SE2",
          "bank_id": "403000",
          "bank_id_code": "GBDSC",
          "name": "Wilfred Jeremiah Owens"
        },
        "charges_information": {
          "bearer_code": "SHAR",
          "sender_charges": [
            {
              "amount": "5.00",
              "currency": "GBP"
            },
            {
              "amount": "10.00",
              "currency": "USD"
            }
          ],
          "receiver_charges_amount": "1.00",
          "receiver_charges_currency": "USD"
        },
        "currency": "GBP",
        "debtor_party": {
          "account_name": "EJ Brown Black",
          "account_number": "GB29XABC10161234567801",
          "account_number_code": "IBAN",
          "address": "10 Debtor Crescent Sourcetown NE1",
          "bank_id": "203301",
          "bank_id_code": "GBDSC",
          "name": "Emelia Jane Brown"
        },
        "end_to_end_reference": "Wil piano Jan",
        "fx": {
          "contract_reference": "FX123",
          "exchange_rate": "2.00000",
          "original_amount": "200.42",
          "original_currency": "USD"
        },
        "numeric_reference": "1002001",
        "payment_id": "123456789012345678",
        "payment_purpose": "Paying for goods/services",
        "payment_scheme": "FPS",
        "payment_type": "Credit",
        "processing_date": "2017-01-18",
        "reference": "Payment for Em's piano lessons",
        "scheme_payment_sub_type": "InternetBanking",
        "scheme_payment_type": "ImmediatePayment",
        "sponsor_party": {
          "account_number": "56781234",
          "bank_id": "123123",
          "bank_id_code": "GBDSC"
        }
      }
    },
```

PUT Methods

- /v1/payment/{paymentID}

Updates a new payment. A example payload would be like the following:

```
{
      "type": "Payment",
      "id": "4ee3a8d8-ca7b-4290-a52c-dd5b6165ec43",
      "version": 0,
      "organisation_id": "743d5b63-8e6f-432e-a8fa-c5d8d2ee5fcb",
      "attributes": {
        "amount": "100.21",
        "beneficiary_party": {
          "account_name": "W Owens",
          "account_number": "31926819",
          "account_number_code": "BBAN",
          "account_type": 0,
          "address": "1 The Beneficiary Localtown SE2",
          "bank_id": "403000",
          "bank_id_code": "GBDSC",
          "name": "Wilfred Jeremiah Owens"
        },
        "charges_information": {
          "bearer_code": "SHAR",
          "sender_charges": [
            {
              "amount": "5.00",
              "currency": "GBP"
            },
            {
              "amount": "10.00",
              "currency": "USD"
            }
          ],
          "receiver_charges_amount": "1.00",
          "receiver_charges_currency": "USD"
        },
        "currency": "GBP",
        "debtor_party": {
          "account_name": "EJ Brown Black",
          "account_number": "GB29XABC10161234567801",
          "account_number_code": "IBAN",
          "address": "10 Debtor Crescent Sourcetown NE1",
          "bank_id": "203301",
          "bank_id_code": "GBDSC",
          "name": "Emelia Jane Brown"
        },
        "end_to_end_reference": "Wil piano Jan",
        "fx": {
          "contract_reference": "FX123",
          "exchange_rate": "2.00000",
          "original_amount": "200.42",
          "original_currency": "USD"
        },
        "numeric_reference": "1002001",
        "payment_id": "123456789012345678",
        "payment_purpose": "Paying for goods/services",
        "payment_scheme": "FPS",
        "payment_type": "Credit",
        "processing_date": "2017-01-18",
        "reference": "Payment for Em's piano lessons",
        "scheme_payment_sub_type": "InternetBanking",
        "scheme_payment_type": "ImmediatePayment",
        "sponsor_party": {
          "account_number": "56781234",
          "bank_id": "123123",
          "bank_id_code": "GBDSC"
        }
      }
    },
```

Delete

- /v1/payment/1

Deletes a payment.

Health

- /health