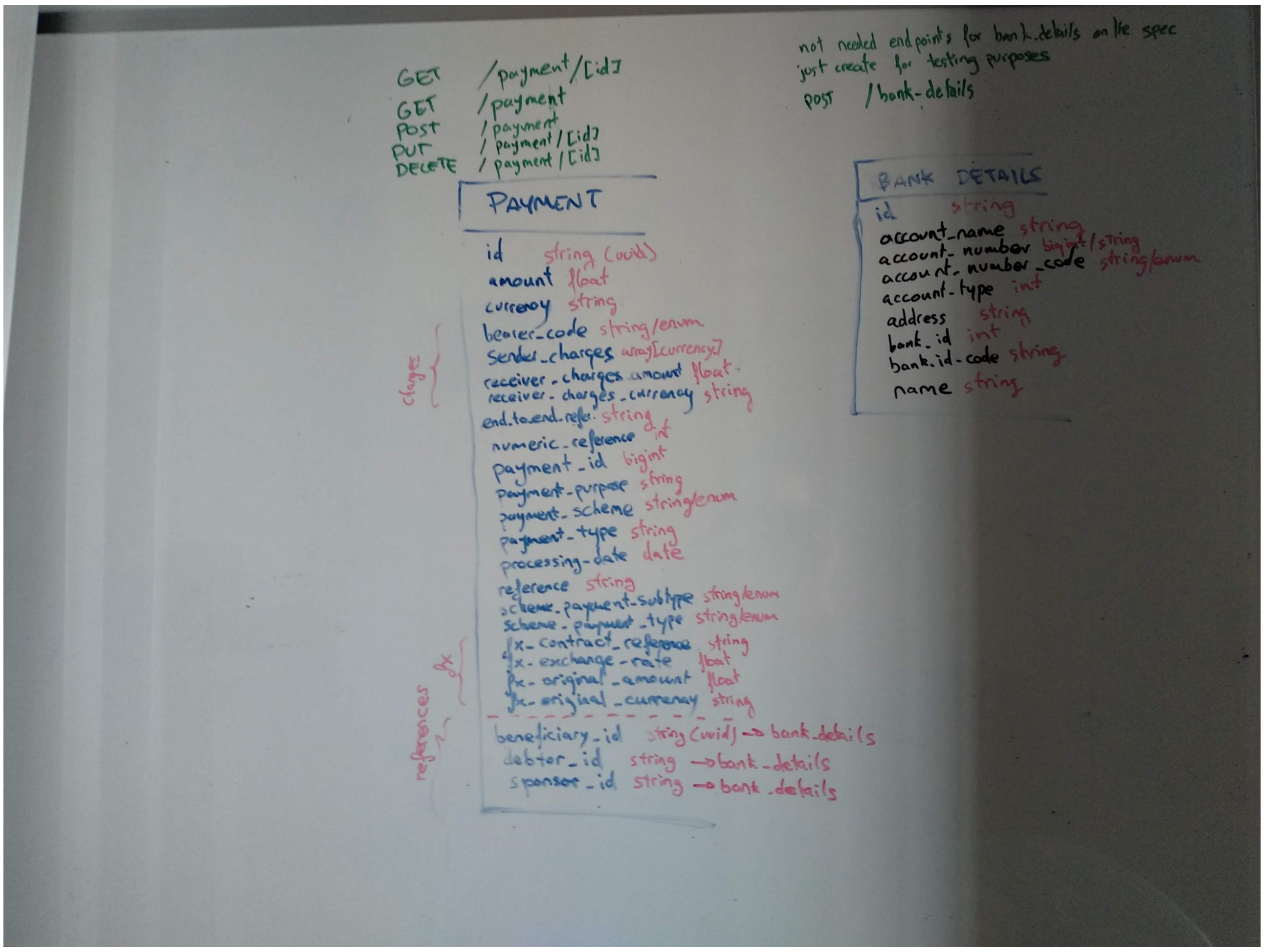
Scheme



```
GET     /payment/[id]
GET     /payment
POST    /payment
PUT     /payment/[id]
DELETE  /payment/[id]
```

not needed endpoints for bank-details on the spec
just create for testing purposes
POST /bank-details

PAYMENT

id	string (uuid)
amount	float
currency	string
bearer_code	string/enum
sender_charges	array[currency]
receiver-charges amount	float
receiver-charges_currency	string
end.to.end.refer.	string
numeric_reference	int
payment_id	bigint
payment-purpose	string
payment-scheme	string/enum
payment-type	string
processing-date	date
reference	string
scheme-payment-subtype	string/enum
scheme-payment-type	string/enum
fx-contract_reference	string
fx-exchange-rate	float
fx-original_amount	float
fx-original_currency	string
beneficiary-id	string (uuid) → bank.details
debtor_id	string → bank-details
sponsor-id	string → bank.details

charges
fx
references

BANK DETAILS

id	string
account_name	string
account-number	bigint/string
account-number_code	string/enum
account-type	int
address	string
bank-id	int
bank.id-code	string
name	string

PAYMENT

Id	String (uuid)
Amount	Float
Currency	String
Bearer_code	String/enum
Sender_charges	Array[money]
Receiver_charges_amout	Float
Receiver_charges_currency	String
End_to_end_reference	String
Numeric_reference	Int
Payment_id	Bigint
Payment_purpose	String
Payment_scheme	String/enum
Payment_type	String
Processing_date	Date
Reference	String
Scheme_payment_subtype	String/enum
Scheme_payment_type	String/enum
Fx_contract_reference	String
Fx_exchange_rate	Float
Fx_original_amount	Float
Fx_original_currency	String
Beneficiary_id	String (uuid)
Debtor_id	String (uuid)
Sponsor_id	String (uuid)

ACCOUNT

Id	String (uuid)
Account_name	String
Account_number	bigint/String
Account_number_code	String/enum
Account_type	Int
Address	String
Bank_id	Int
Bank_id_code	String
Name	String

Endpoints

PAYMENT

- Get a single payment by id

GET /payment/[id]

- Get a list of payments

GET /payment

- Create a new payment

POST /payment

- Edit an existing payment

PUT /payment/[id]

- Delete a payment

DELETE /payment/[id]

Assumptions and comments

I noticed the example given is using a format similar to Jsonapi, we would need a serializer to translate the database models into the Jsonapi format, but this is an implementation detail.

Jsonapi encourages to add external entities as references in the Json output, however I noticed in this example that everything is inside attributes, so I'll keep this format to match the example on the implementation.

I considered version and organisation_id (maybe some kind of internal client id?) to be api versioning information, therefore I did not include them in the models.

Beneficiary, debtor and sponsor parties are all bank detail references, as they share msot of the attributes and can be reused across different payments.

To avoid payment conflicts, we could allow creation on accounts and not updates. If a person changes is bank details, a new entity is create. Optionally, we could add a flag to mark outdated bank details.

Foreigh exchange is specific for every payment, and a such it doesn't make sense to store as a separate resource.

I decided to store charges as an array of json to keep currency and amount separate. I considered this is not going to be queryeable otherwise I would store them as an auxiliary table.

Internal ids are going to be stored as UUIDs in the database (as the one in the payment example)

There was no requirement for a accounts (beneficiary, debtor or sponsor external references) api, so I have just added a POST for testing purposes.

In principle, I consider all fields in Payment are mandatory.

I am not familiar with the meaning of some fields like the difference between reference or end_to_end_reference, or why there are multiple sender_charges in different currencies; so I decided to keep the model to respect the format.

Money parameters (amount in float and currency in string) could be replaced by a library like https://github.com/JodaOrg/joda-money . I’ve used it before in Scala and it works good, only a few conversions needs to be done to translate into and from the database and the api.