Table of contents

Architecture Design

The micro-services architecture is really important for applications, but I select a standalone application, because provides more facilities of run, test and show you the

requirements, with more than one components I need to share some code for the purpose of reduce duplicated code and more time to connect the services and the problem is easier than that.

Code design choices

I've use many tools and I'll describe it:
Java 11: this is the latest LTS, that means that the components are stable and long supported.
TDD with JUnit5: junit 5 is clear to use and has many features to allow better testing, with TDD I've used some tests to try quickly if my controllers and logic was working ok while I was programming.
WireMock: I've consumed an external service for currency rates information, this was mocked in tests for testing isolation.
Drools: the logic of the application can be change with configuration files that compiles at application startup, this a common feature to change the logic quickly even in prod environments
Change of currencies API: The API https://api.exchangeratesapi.io/latest?base=USD is obsolete and isn't working without api-key, so I've create one to use the newest, but the base option of the API is only available for paid users so, I don't use it, instead I've mock a similar one with postman.
WebFlux: Using a different approach to solve problems shows you my skill in learning, and this reactive programming is useful for applications with many concurrent requests with non-blocking connections. This is more difficult to implement because the information and the community can be larger but it's really profitable.
H2: this is require to execute in standalone mode
PostgreSQL: the application currently support postgres database with the same logic, it can be ran with **docker-compose**
Format google java code: I've use the formatter of google for java code, this is important in quality (on clean code terms) and readability
Logger: I've used the native logging tool and not the Apache Log4j only because are an included feature in spring boot
r2dbc: instead of jpa or jdbc, because they have support to webflux implementation
jacoco: to report coverage to sonarqube
I expose more resources and methods to the API and change one because on my design accounts with others Currencies are supported.
I've been using Custom error classes and I've a single class to handle it.

Quality, maintainability and extensibility

Quality was check with sonarqube and intellij inspector tools, to prevent missing errors, the ttd testing using the behavior of interface is incredible useful to ensure easy refactoring supporting the maintainability feature, and is so extensible because the spring framework community has a lot of plugins to adapt the code to new challenges with few changes.

API Design

I've use some patterns to describe the work

Using semantic version to URI

Using base path indicating the mayor version, changing between apis with URL using an api gateway:
/v1

Using entity-collection

Get Customer entity:
GET /v1/customers/:idCustomer
Get Customer collection:
GET /v1/customers
Create a transaction:
POST /v1/transactions

Using master-slave design pattern accounts is slave of customers

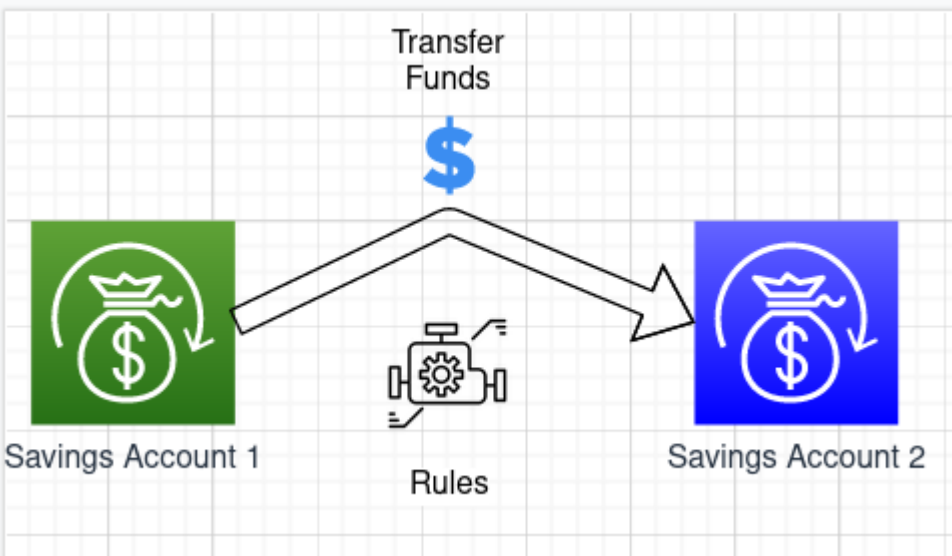
Get single account:
GET /v1/customers/:idCustomer/accounts/:idAccount

Get all accounts of customer:
GET /v1/customers/:idCustomer/accounts

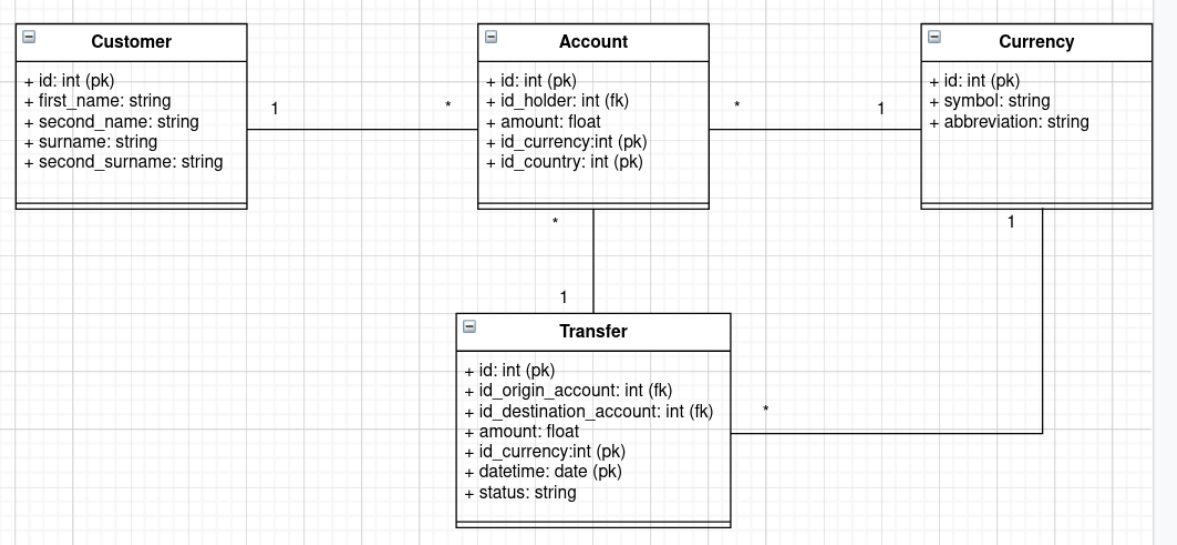
Using command pattern

Get a single account with a specific message, the command pattern allows POST to retrieve information, this is used only because the challenge specifies this:
POST /v1/customers/130303/retrieve-account

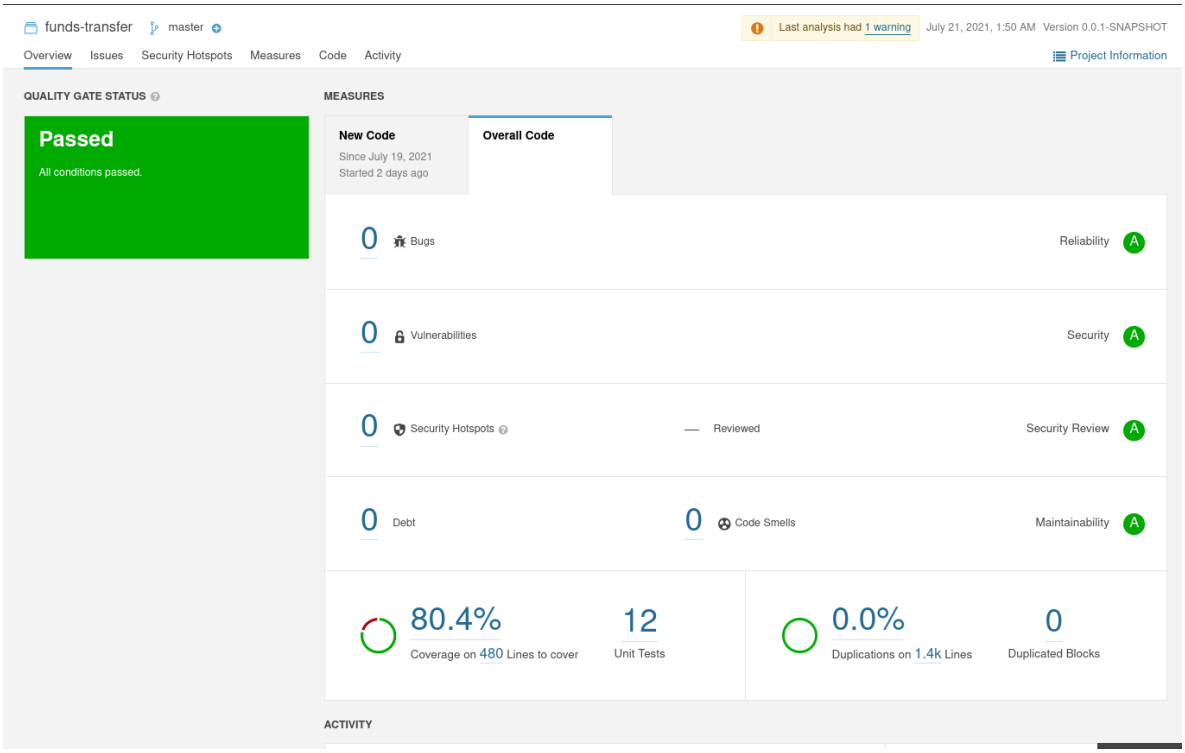
Problem understanding

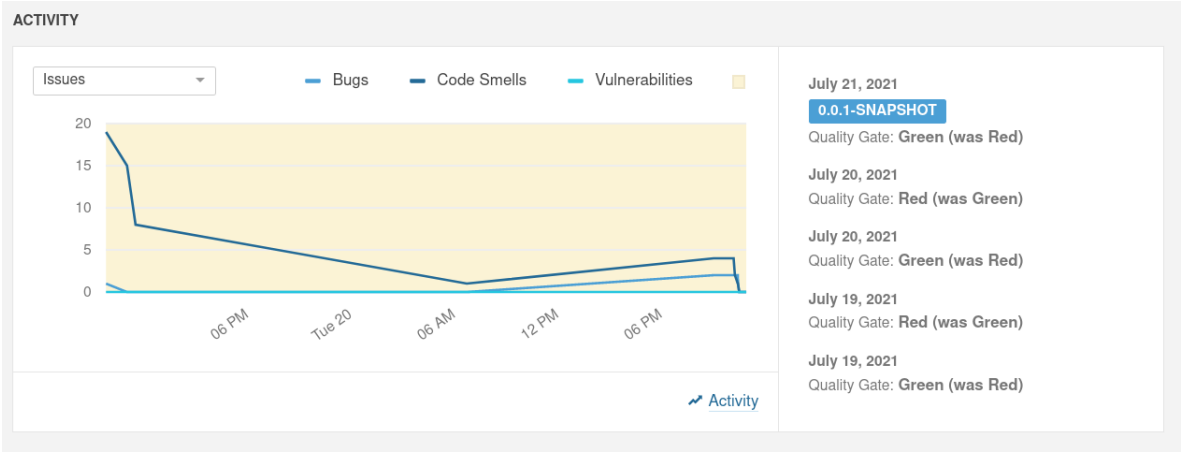


Entities (E-R)

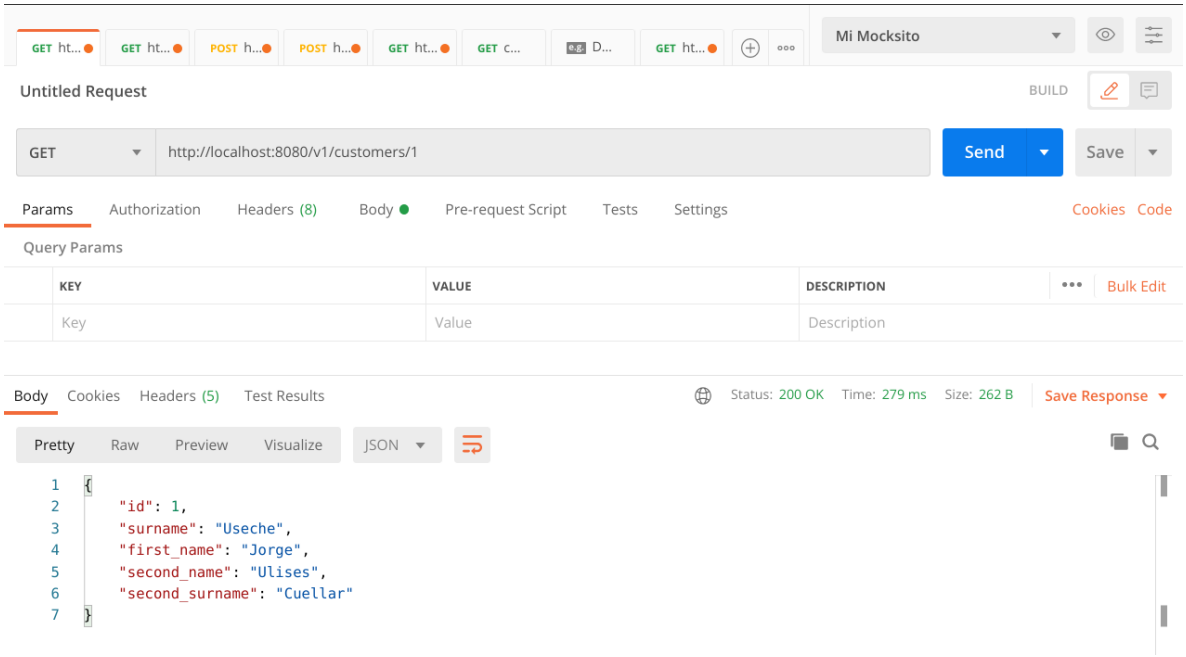


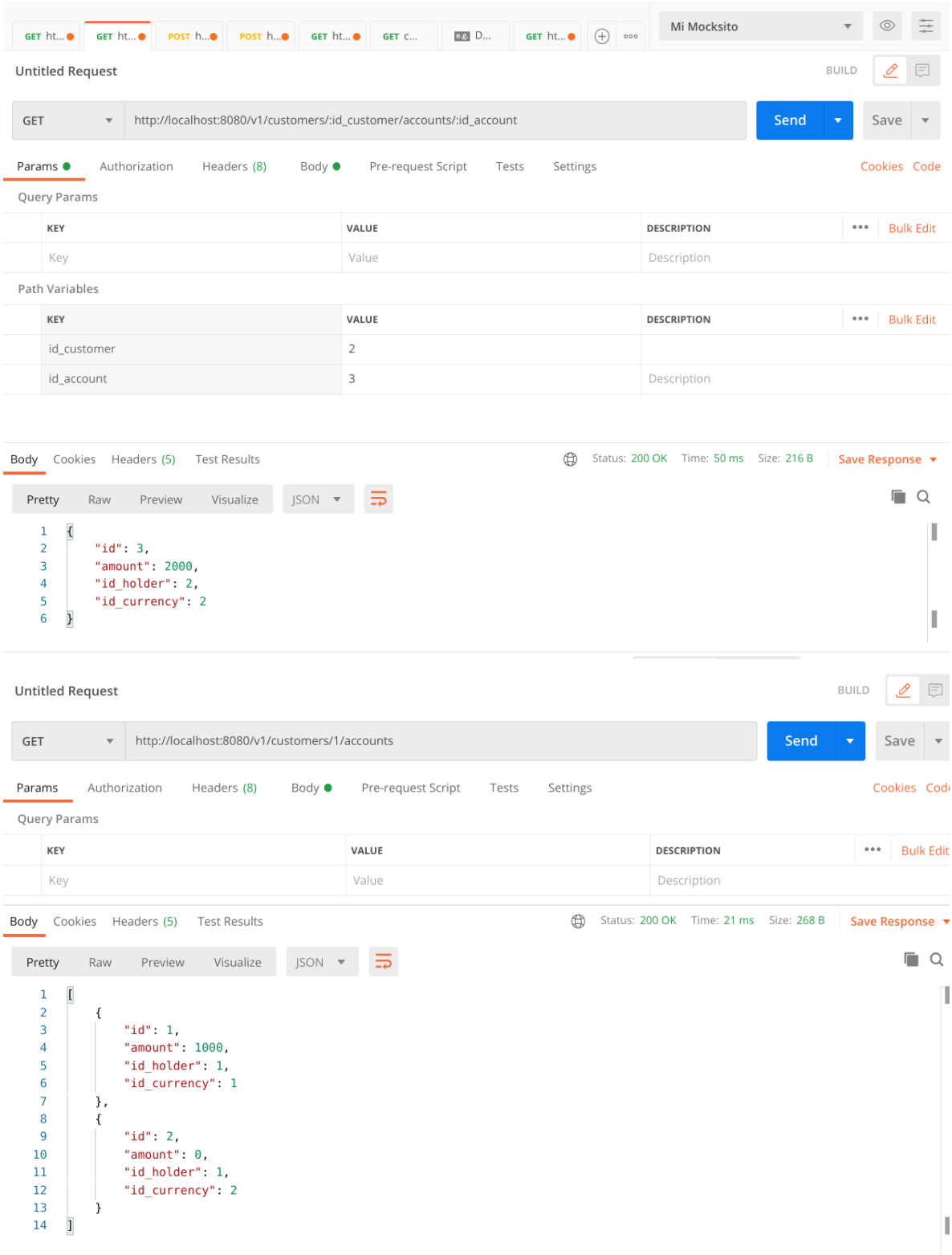
SonarQube report

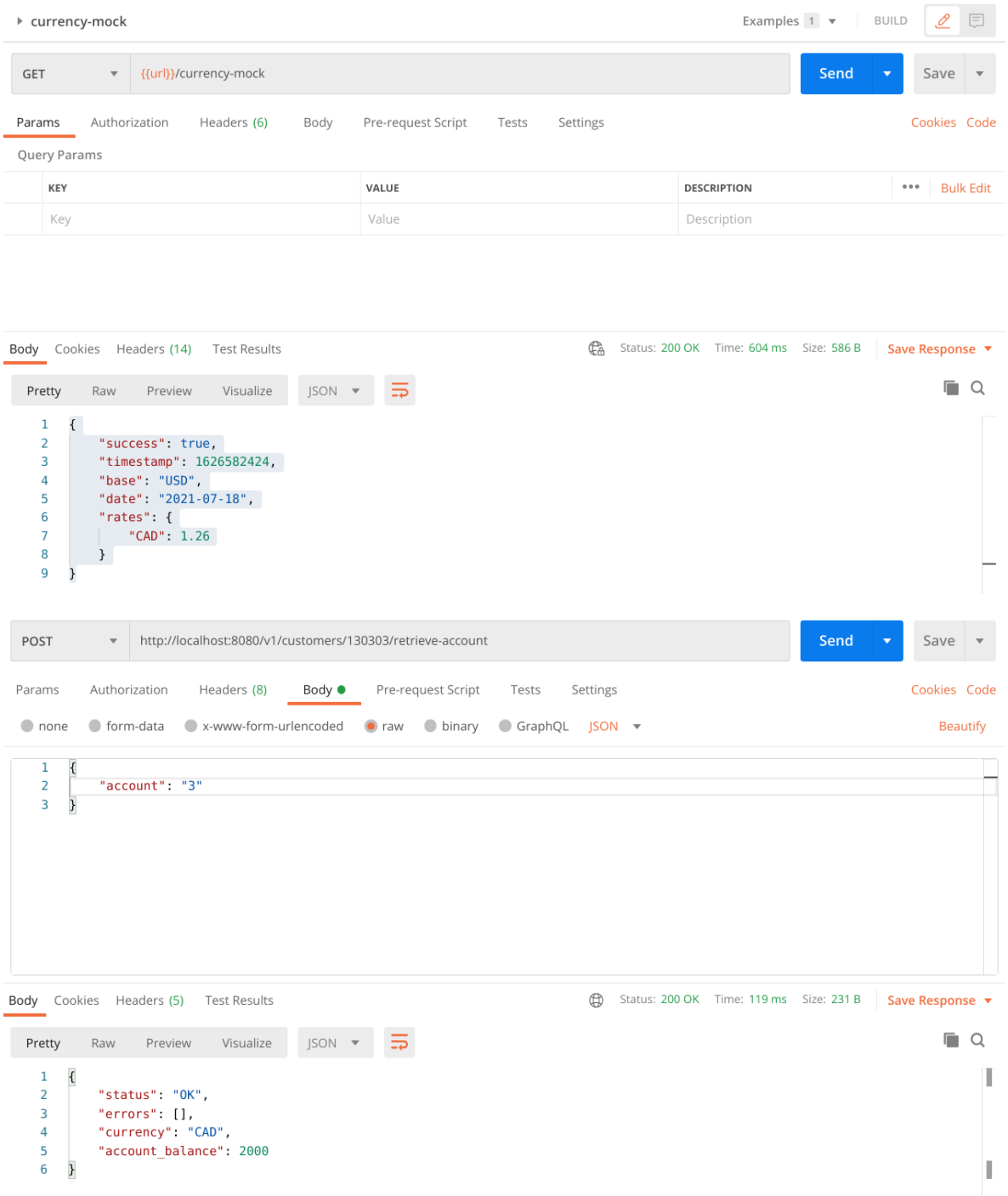


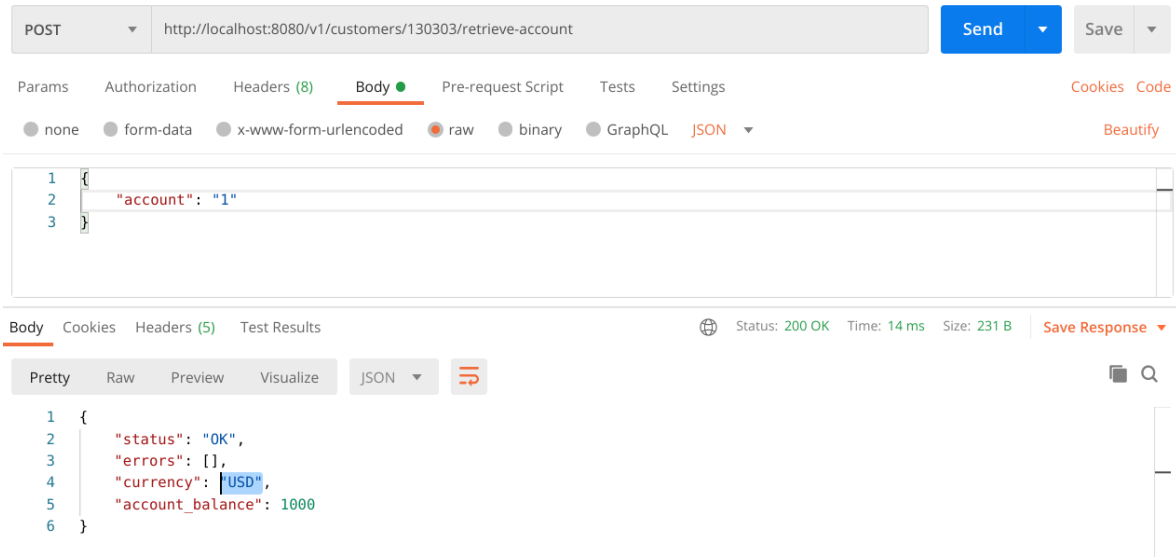


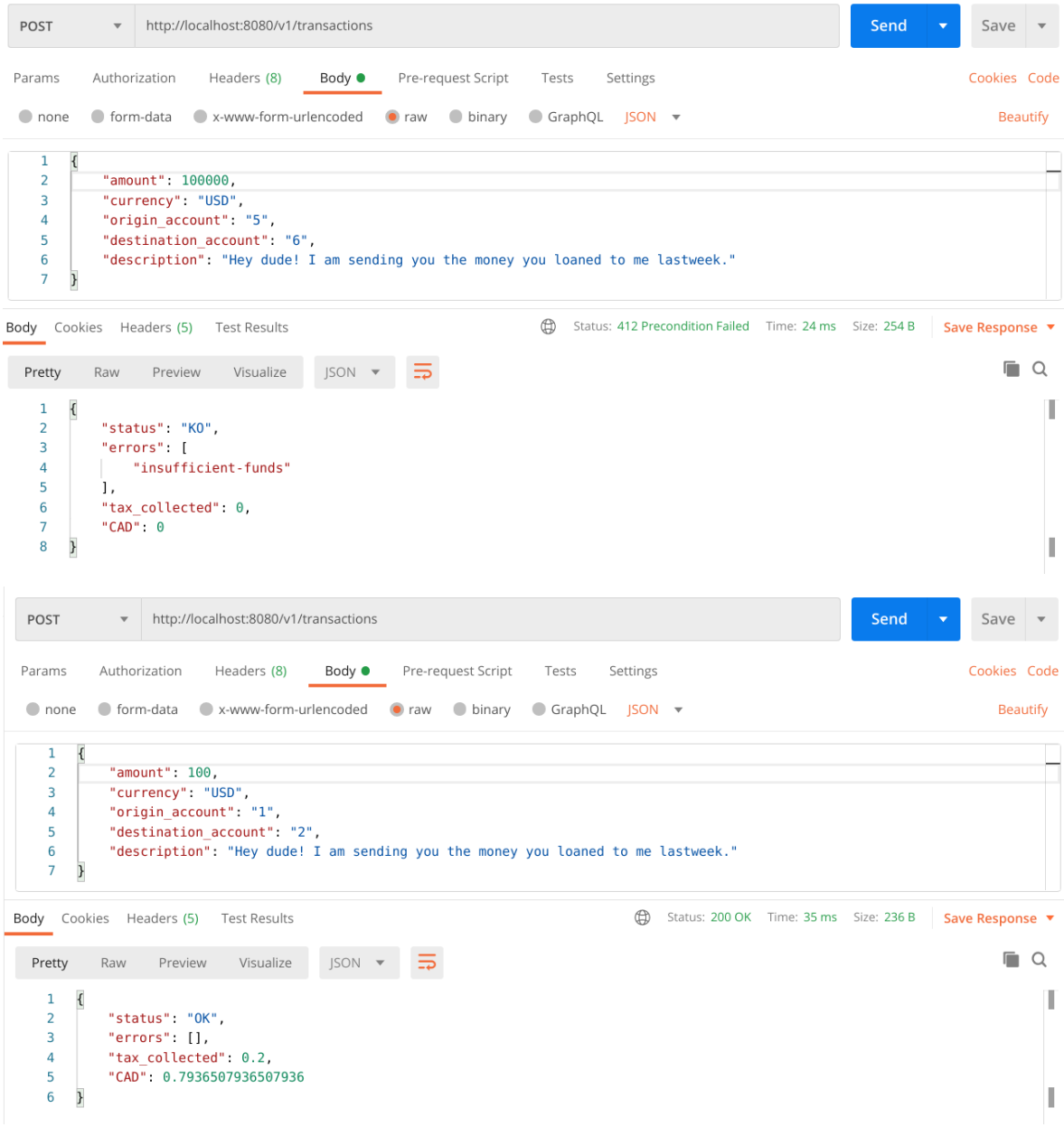
API consume in postman



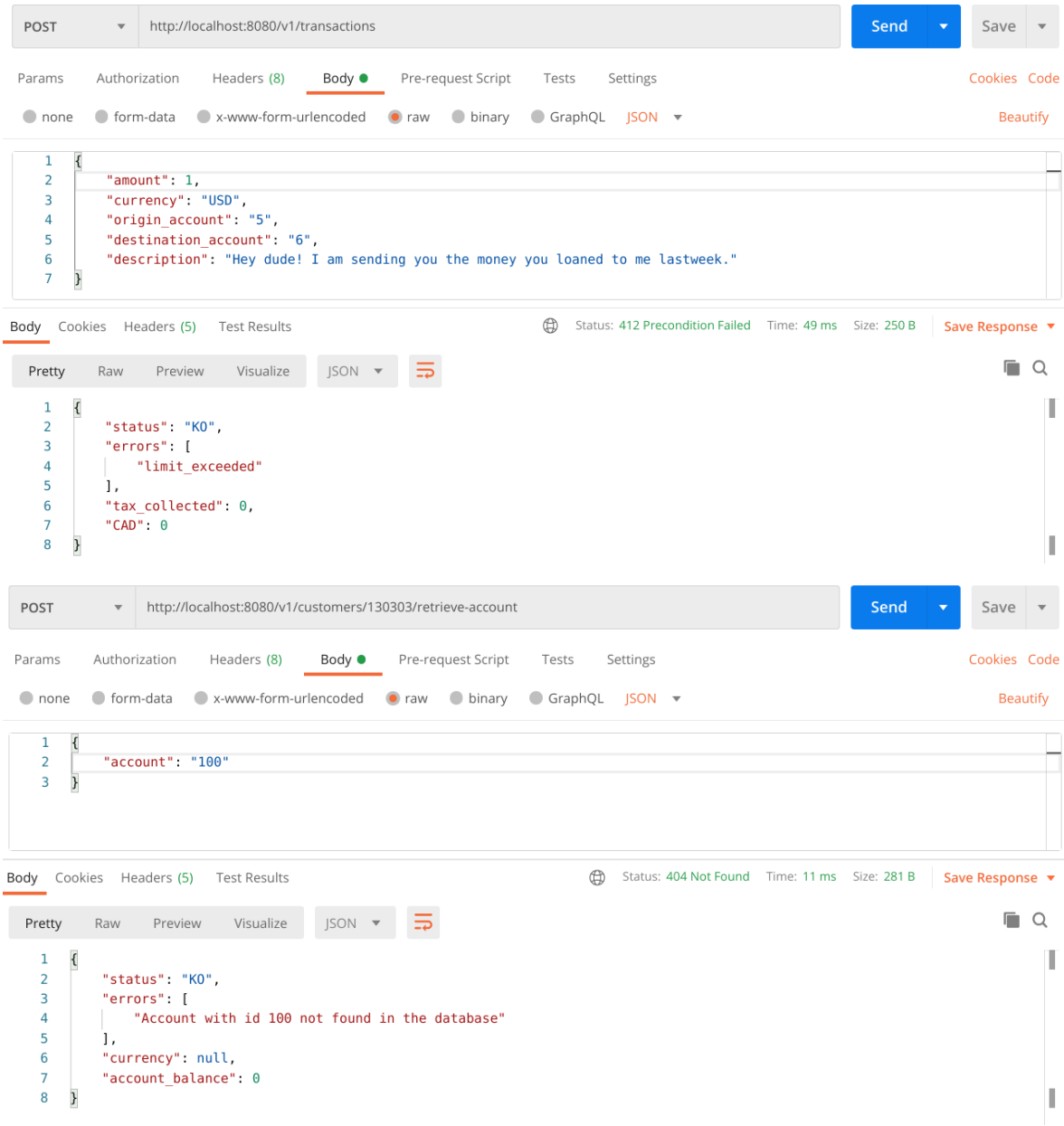

Mi Mocksito
Untitled Request
BUILD
GET
http://localhost:8080/v1/customers/:id_customer/accounts/:id_account
Send
Save
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
Code
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
Key
Value
Description
Path Variables
id_customer
2
id_account
3
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 50 ms
Size: 216 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"id": 3,
"amount": 2000,
"id_holder": 2,
"id_currency": 2
}
Untitled Request
BUILD
GET
http://localhost:8080/v1/customers/1/accounts
Send
Save
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
Key
Value
Description
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 21 ms
Size: 268 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
[
{
"id": 1,
"amount": 1000,
"id_holder": 1,
"id_currency": 1
},
{
"id": 2,
"amount": 0,
"id_holder": 1,
"id_currency": 2
}
]


currency-mock
Examples 1
BUILD
GET
{{url}}/currency-mock
Send
Save
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
Code
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
Key
Value
Description
Body
Cookies
Headers (14)
Test Results
Status: 200 OK
Time: 604 ms
Size: 586 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"success": true,
"timestamp": 1626582424,
"base": "USD",
"date": "2021-07-18",
"rates": {
"CAD": 1.26
}
}
POST
http://localhost:8080/v1/customers/130303/retrieve-account
Send
Save
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
Code
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"account": "3"
}
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 119 ms
Size: 231 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"status": "OK",
"errors": [],
"currency": "CAD",
"account_balance": 2000
}


POST
http://localhost:8080/v1/customers/130303/retrieve-account
Send
Save
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
Code
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"account": "1"
}
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 14 ms
Size: 231 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"status": "OK",
"errors": [],
"currency": "USD",
"account_balance": 1000
}


POST
http://localhost:8080/v1/transactions
Send
Save
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
Code
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"amount": 100000,
"currency": "USD",
"origin_account": "5",
"destination_account": "6",
"description": "Hey dude! I am sending you the money you loaned to me lastweek."
}
Body
Cookies
Headers (5)
Test Results
Status: 412 Precondition Failed
Time: 24 ms
Size: 254 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"status": "KO",
"errors": [
"insufficient-funds"
],
"tax_collected": 0,
"CAD": 0
}

POST
http://localhost:8080/v1/transactions
Send
Save
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
Code
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"amount": 100,
"currency": "USD",
"origin_account": "1",
"destination_account": "2",
"description": "Hey dude! I am sending you the money you loaned to me lastweek."
}
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 35 ms
Size: 236 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"status": "OK",
"errors": [],
"tax_collected": 0.2,
"CAD": 0.7936507936507936
}

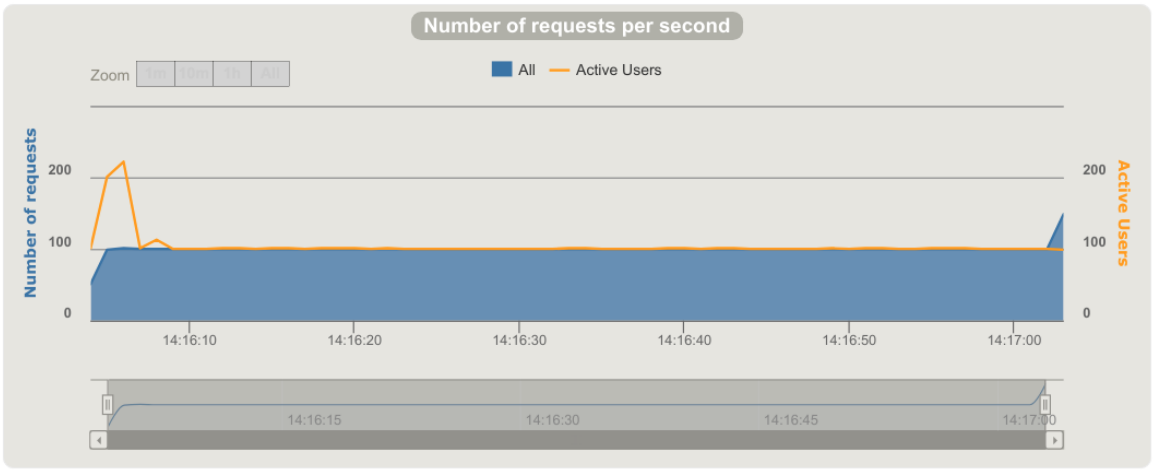

POST
http://localhost:8080/v1/transactions
Send
Save
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
Code
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"amount": 1,
"currency": "USD",
"origin_account": "5",
"destination_account": "6",
"description": "Hey dude! I am sending you the money you loaned to me lastweek."
}
Body
Cookies
Headers (5)
Test Results
Status: 412 Precondition Failed
Time: 49 ms
Size: 250 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"status": "KO",
"errors": [
"limit_exceeded"
],
"tax_collected": 0,
"CAD": 0
}

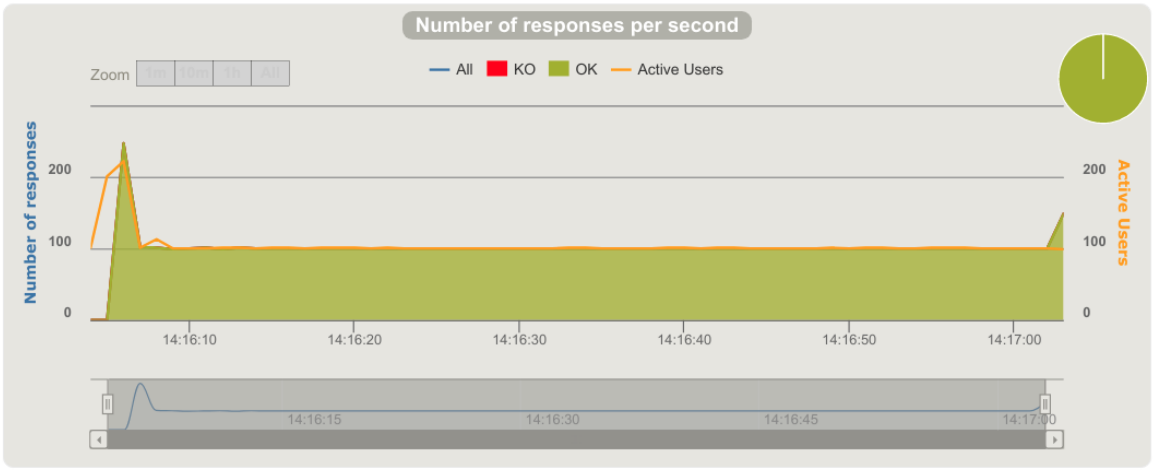
POST
http://localhost:8080/v1/customers/130303/retrieve-account
Send
Save
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
Code
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"account": "100"
}
Body
Cookies
Headers (5)
Test Results
Status: 404 Not Found
Time: 11 ms
Size: 281 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"status": "KO",
"errors": [
"Account with id 100 not found in the database"
],
"currency": null,
"account_balance": 0
}

Gatling testing load and stress (performance in concurrent request)

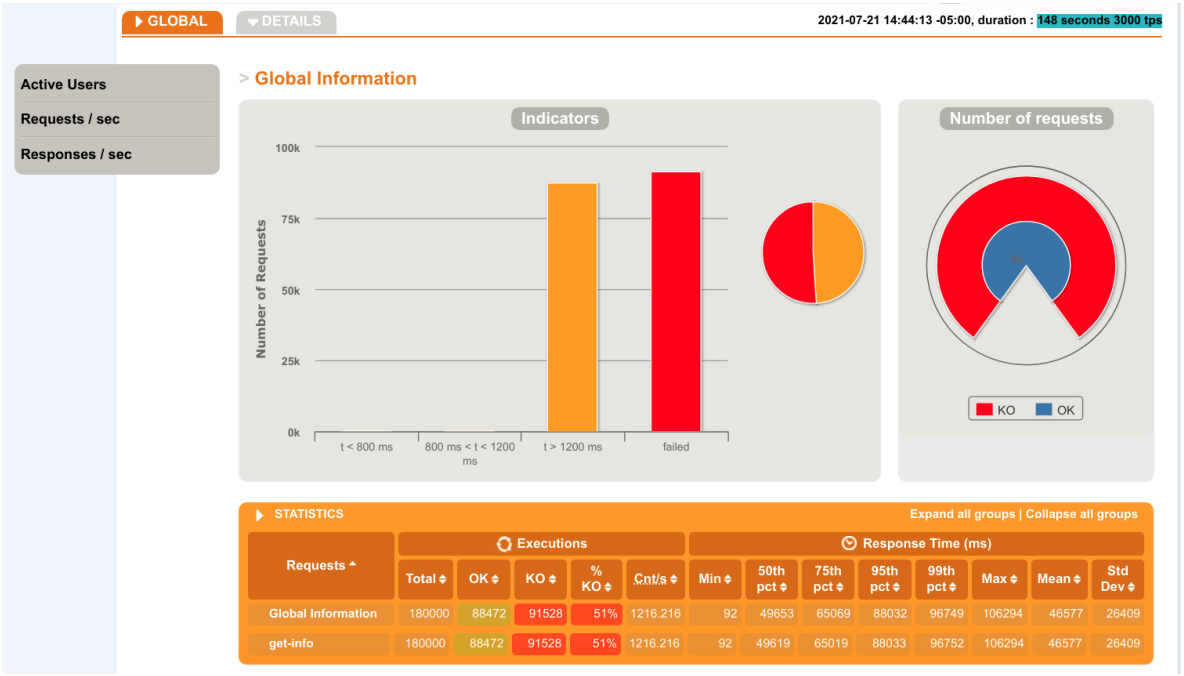
PATH:
http://localhost:8080/v1/customers/130303/retrieve-account
GET

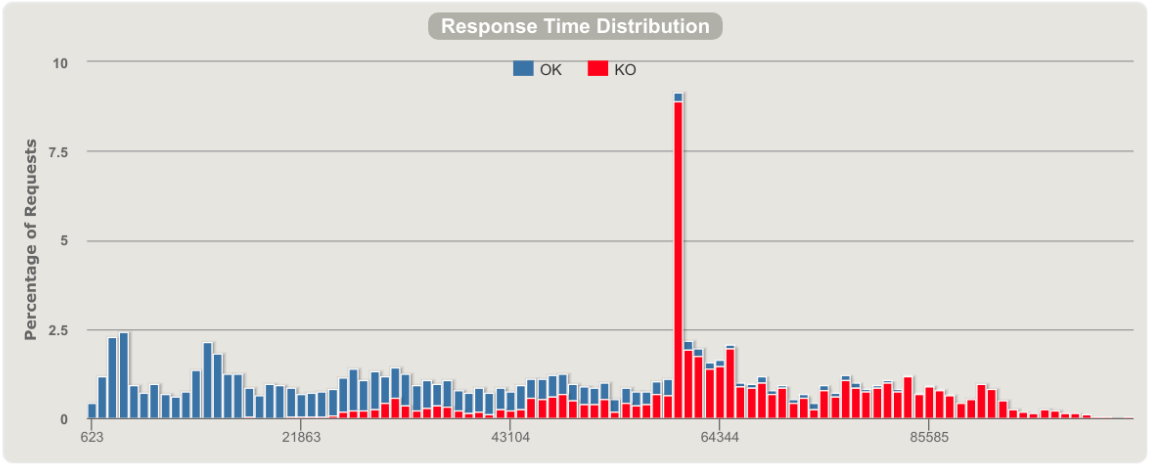
Load test 100 rps 1 node->

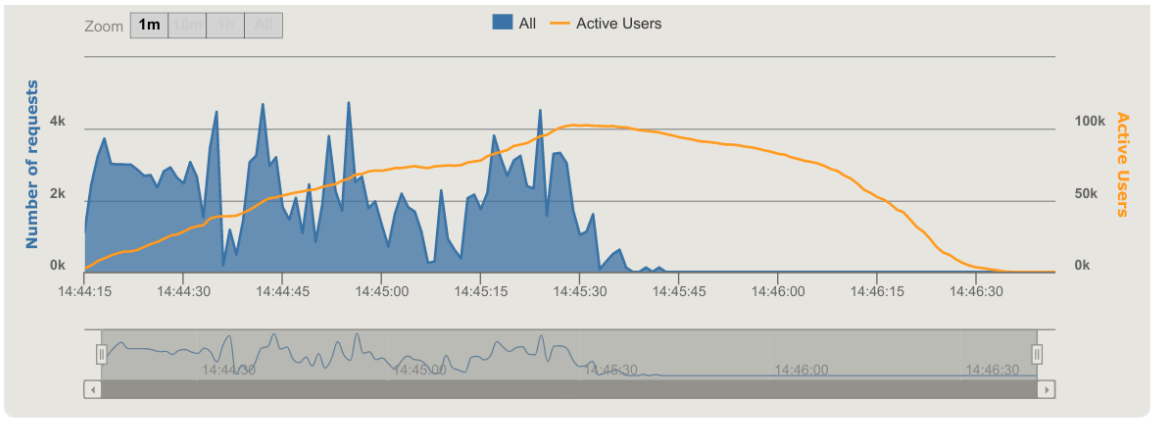

Number of requests per second
Zoom
1m
10m
1h
All
All
Active Users
Number of requests
Active Users
200
100
0
14:16:10
14:16:20
14:16:30
14:16:40
14:16:50
14:17:00
14:16:15
14:16:30
14:16:45
14:17:00

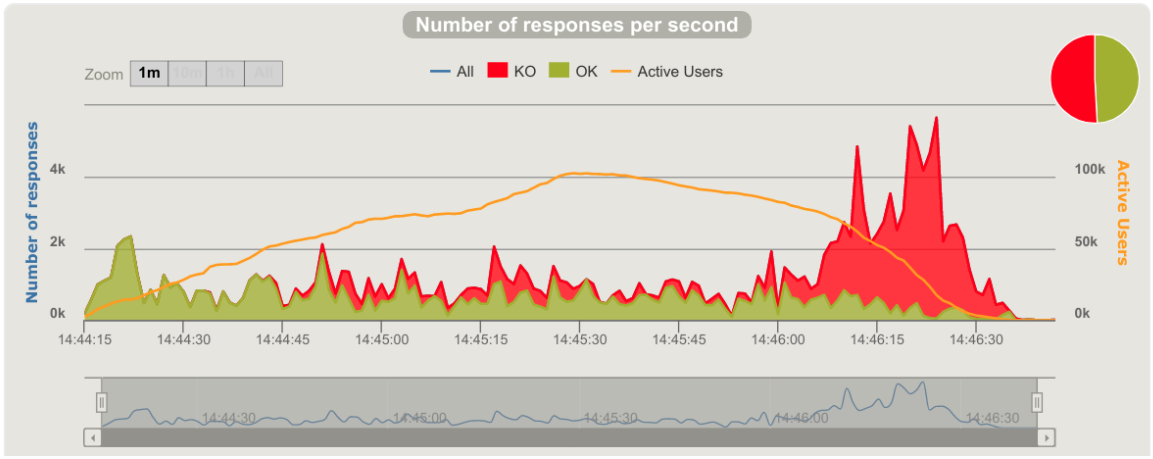

Number of responses per second
Zoom
1m
10m
1h
All
All
KO
OK
Active Users
Number of responses
Active Users
200
100
0
14:16:10
14:16:20
14:16:30
14:16:40
14:16:50
14:17:00
14:16:15
14:16:30
14:16:45
14:17:00

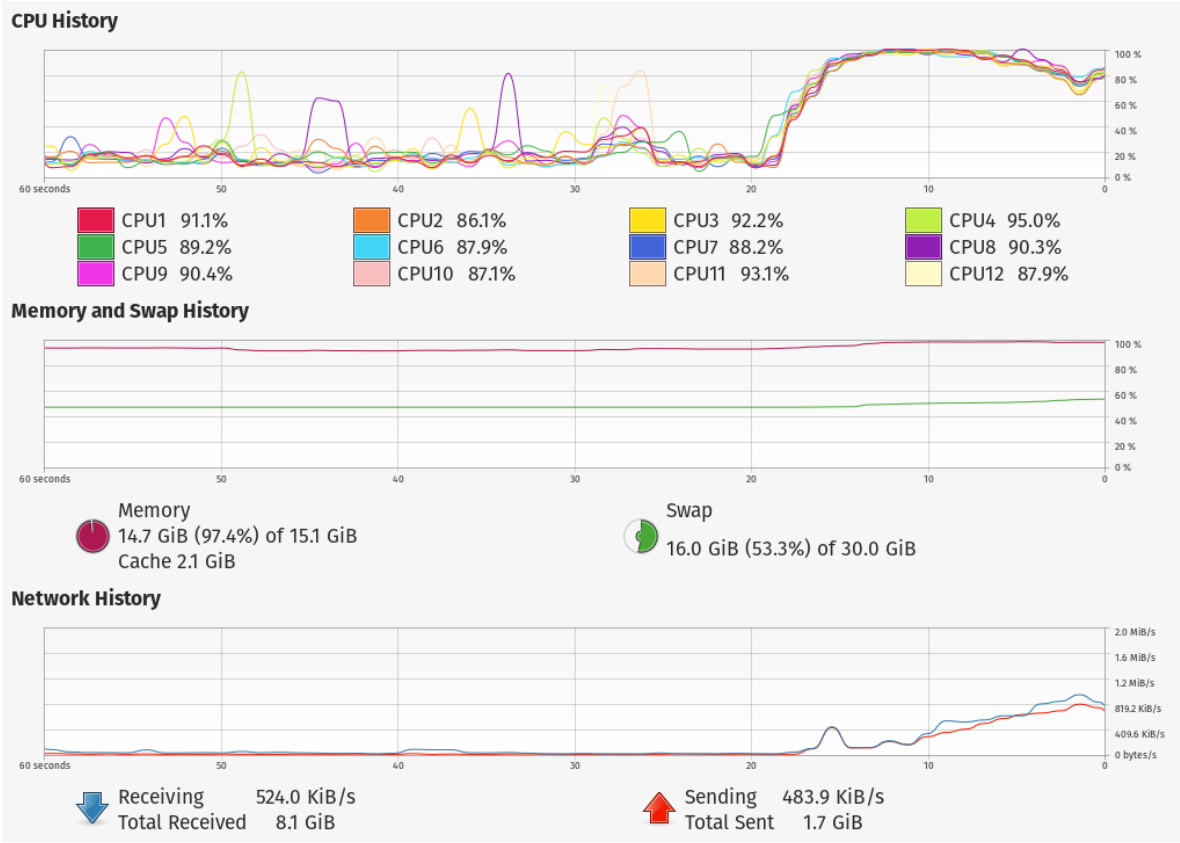
Stress test 3000 rps 1 node ->






Zoom
1m
10m
1h
All
All
Active Users
Number of requests
Active Users
4k
2k
0k
100k
50k
0k
14:44:15
14:44:30
14:44:45
14:45:00
14:45:15
14:45:30
14:45:45
14:46:00
14:46:15
14:46:30


Number of responses per second
Zoom
1m
10m
1h
All
All
KO
OK
Active Users
Number of responses
Active Users
4k
2k
0k
100k
50k
0k
14:44:15
14:44:30
14:44:45
14:45:00
14:45:15
14:45:30
14:45:45
14:46:00
14:46:15
14:46:30



More than request brokes the computer OS limits with Java problems like this

```
java.lang.OutOfMemoryError: Java heap space
Dumping heap to java_pid725443.hprof ...
Heap dump file created [1769915655 bytes in 76.692 secs]
```

Or this:

```
---- Response Time Distribution ------------------------------------------------
> t < 800 ms                                            494 (  0%)
> 800 ms < t < 1200 ms                                  324 (  0%)
> t > 1200 ms                                         87654 ( 49%)
> failed                                              91528 ( 51%)
---- Errors --------------------------------------------------------------------
> i.n.c.ConnectTimeoutException: connection timed out: localhost  31708 (34.64%)
/0:0:0:0:0:0:0:1:8080
> i.g.h.c.i.RequestTimeoutException: Request timeout after 60000  30015 (32.79%)
 ms
> i.g.h.c.i.RequestTimeoutException: Request timeout to localhos  16341 (17.85%)
t/0:0:0:0:0:0:0:1:8080 after 60000 ms
> i.g.h.c.i.RequestTimeoutException: Request timeout to localhos  12829 (14.02%)
t/127.0.0.1:8080 after 60000 ms
> j.i.IOException: Premature close                                  635 ( 0.69%)
================================================================================
```

Or this:

```
as thrown by a user handler's exceptionCaught() method while handling the follow
ing exception:
java.lang.OutOfMemoryError: Java heap space
14:35:19.780 [WARN ] i.n.u.c.SingleThreadEventExecutor - An event executor termi
nated with non-empty task queue (33276)
14:35:19.780 [WARN ] i.n.u.c.SingleThreadEventExecutor - An event executor termi
nated with non-empty task queue (32613)
14:35:19.784 [WARN ] i.n.c.AbstractChannelHandlerContext - An exception 'java.la
ng.OutOfMemoryError: Java heap space' [enable DEBUG level for full stacktrace] w
as thrown by a user handler's exceptionCaught() method while handling the follow
ing exception:
java.lang.OutOfMemoryError: Java heap space
14:35:19.783 [WARN ] i.n.c.AbstractChannelHandlerContext - Failed to mark a prom
ise as failure because it has succeeded already: DefaultChannelPromise@3253ef2a(
success)
java.lang.OutOfMemoryError: Java heap space
Exception in thread "gatling-1-14" java.lang.OutOfMemoryError: Java heap space
Exception in thread "gatling-1-24" java.lang.OutOfMemoryError: Java heap space
```

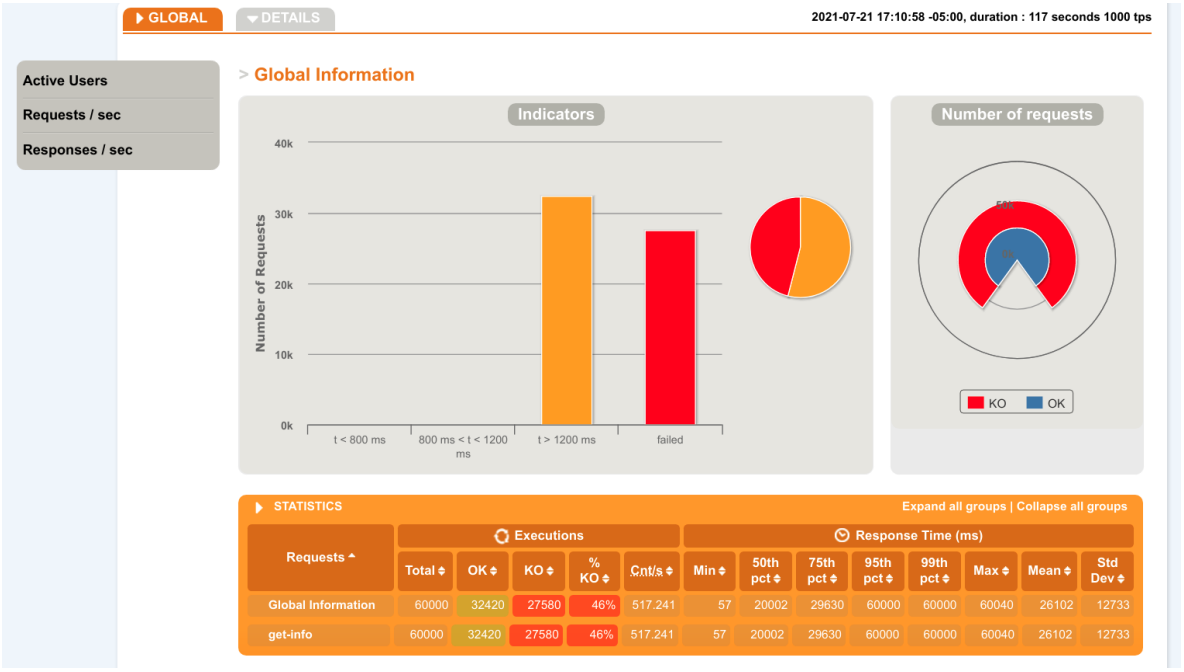
PATH: http://localhost:8080/v1/transactions POST

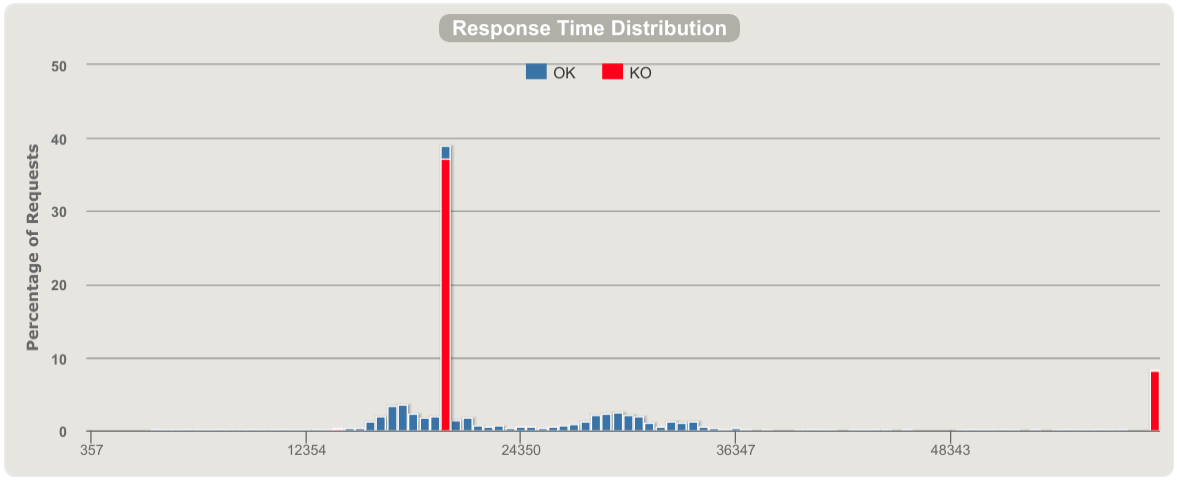
Load test 100 tps 1 node ->

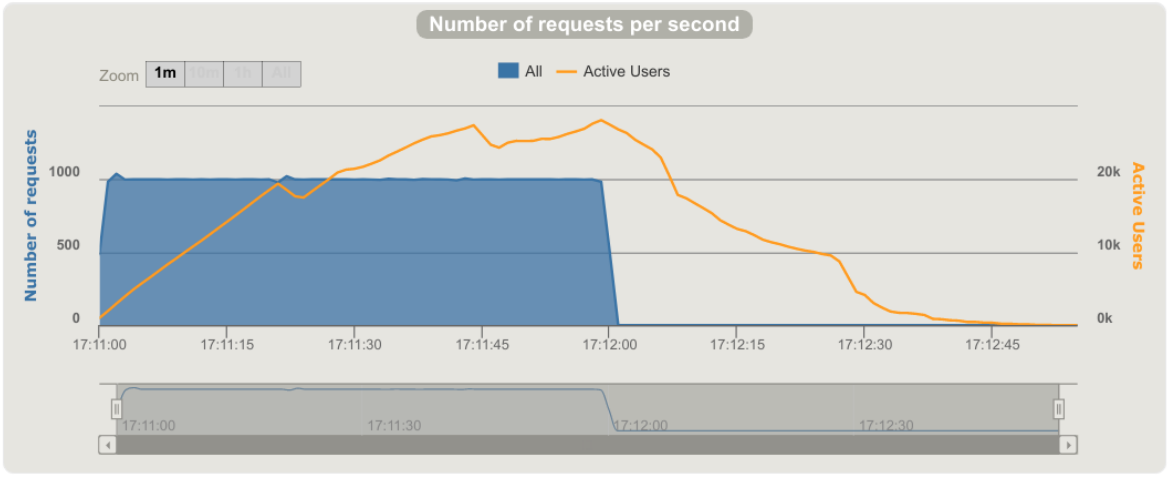
STATISTICS — Expand all groups | Collapse all groups

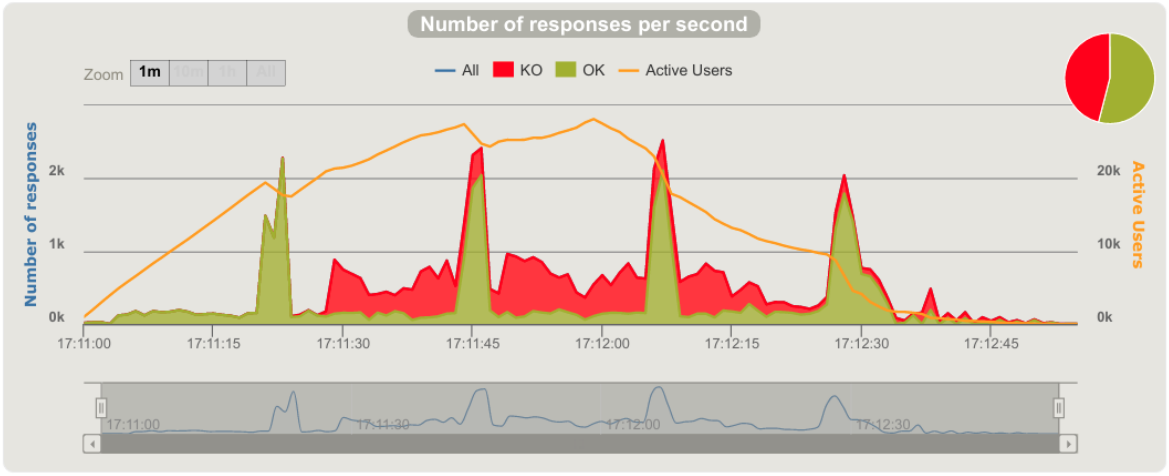
Requests	Executions					Response Time (ms)							
	Total	OK	KO	% KO	Cnt/s	Min	50th pct	75th pct	95th pct	99th pct	Max	Mean	Std Dev
Global Information	6000	5996	4	0%	100	4	6	8	14	17	58	7	3
get-info	6000	5996	4	0%	100	4	6	8	14	17	58	7	3

Stress test 1000 tps 1 node ->

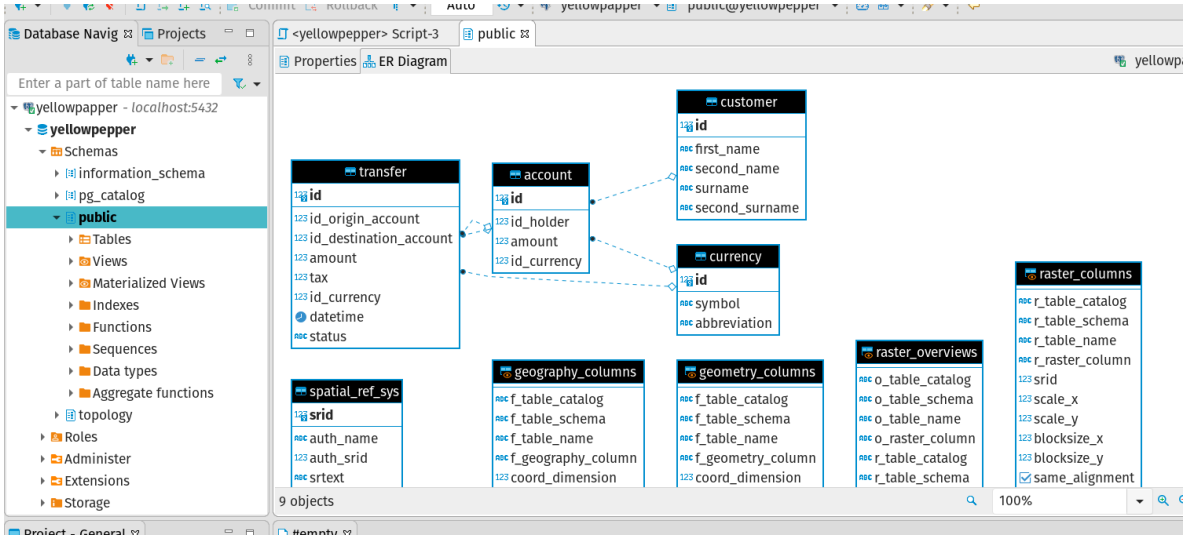


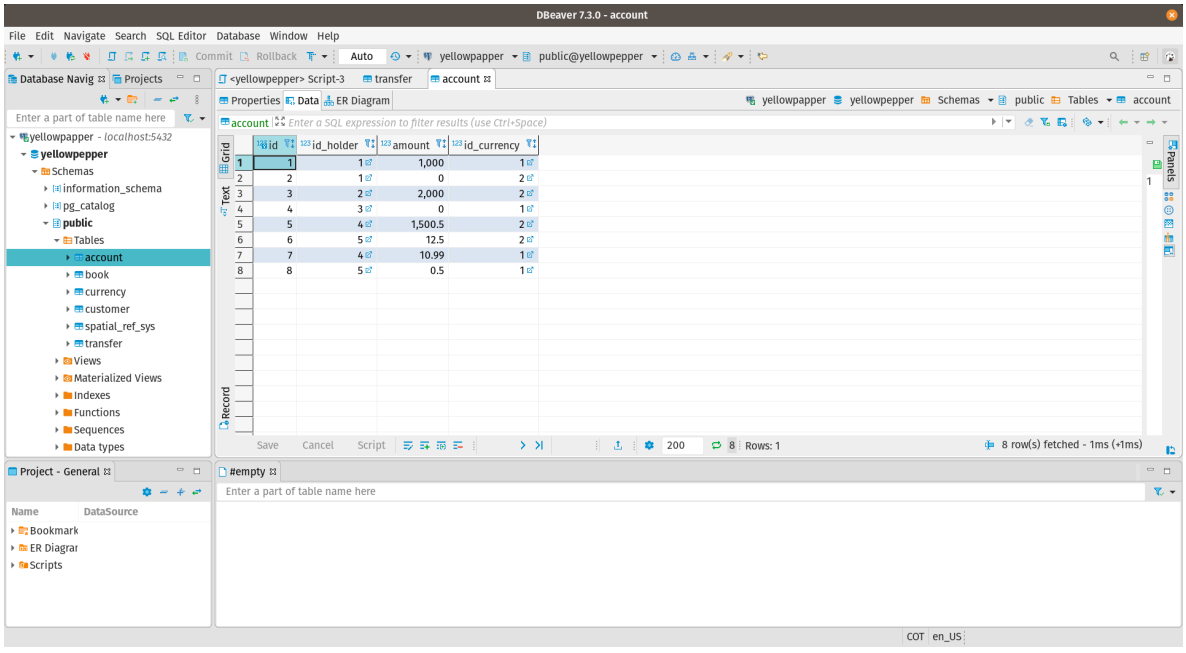


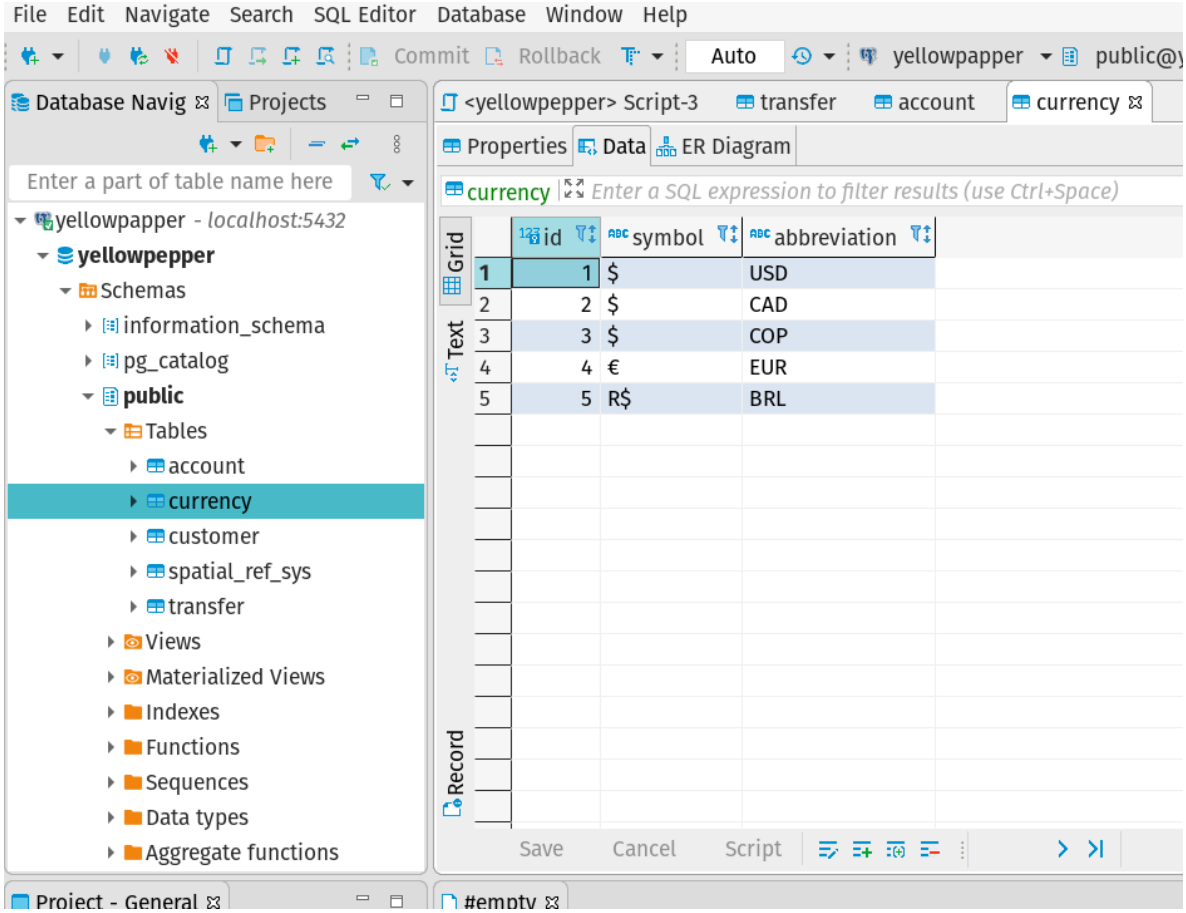

Number of requests per second
Zoom
1m
All
Active Users
Number of requests
1000
500
0
20k
10k
0k
Active Users
17:11:00
17:11:15
17:11:30
17:11:45
17:12:00
17:12:15
17:12:30
17:12:45

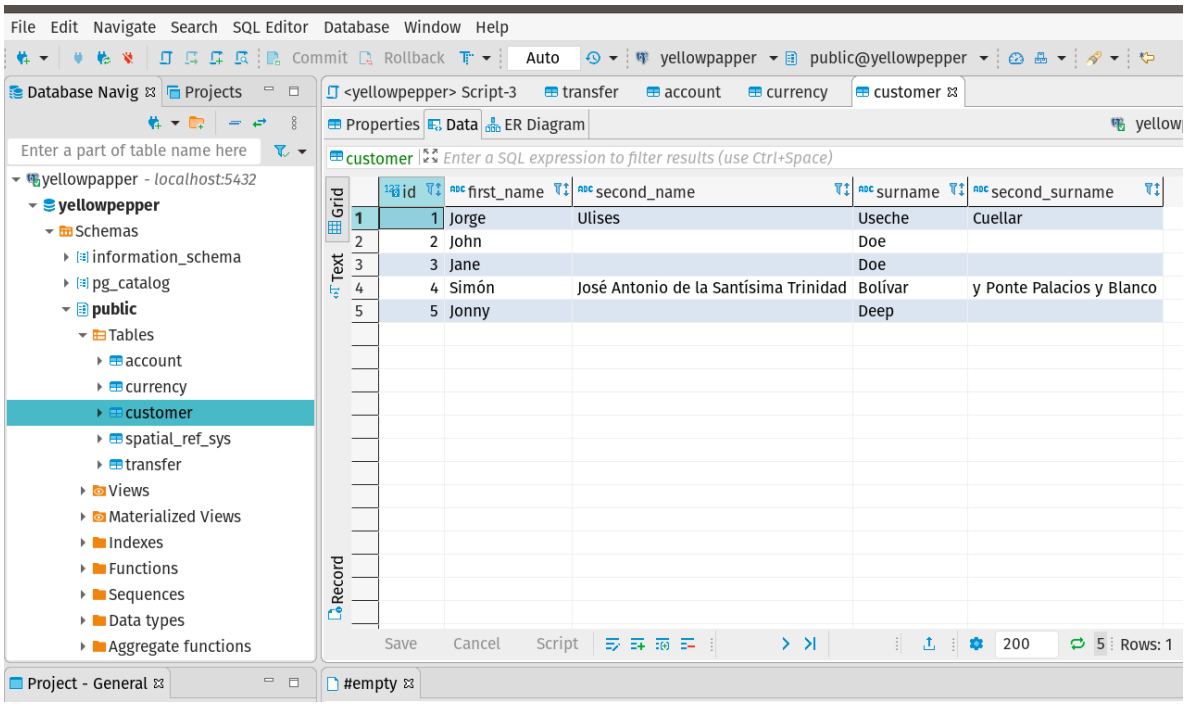

Number of responses per second
Zoom
1m
All
KO
OK
Active Users
Number of responses
2k
1k
0k
20k
10k
0k
Active Users
17:11:00
17:11:15
17:11:30
17:11:45
17:12:00
17:12:15
17:12:30
17:12:45

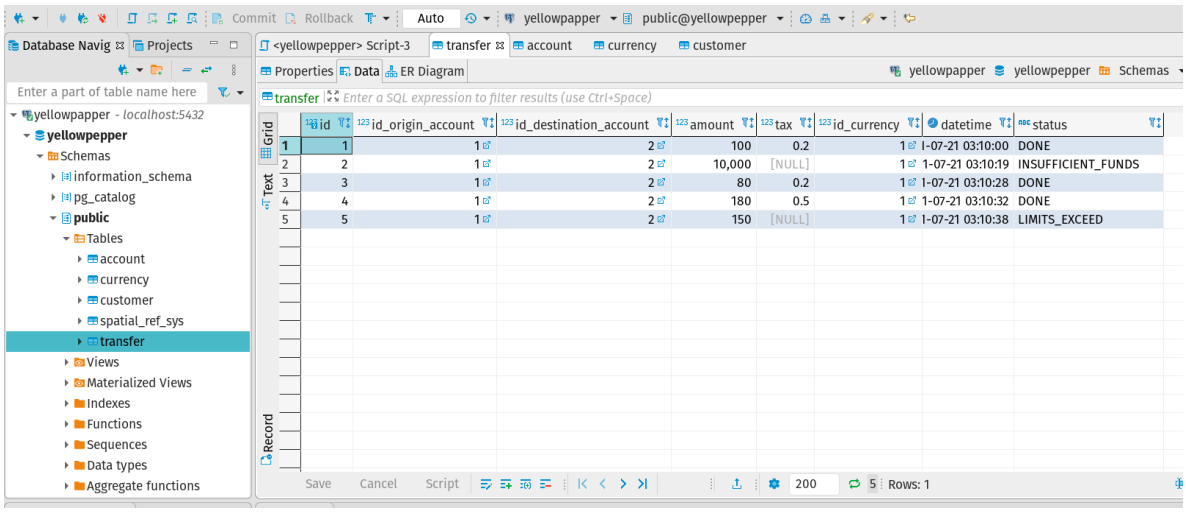
Data in PostgreSQL





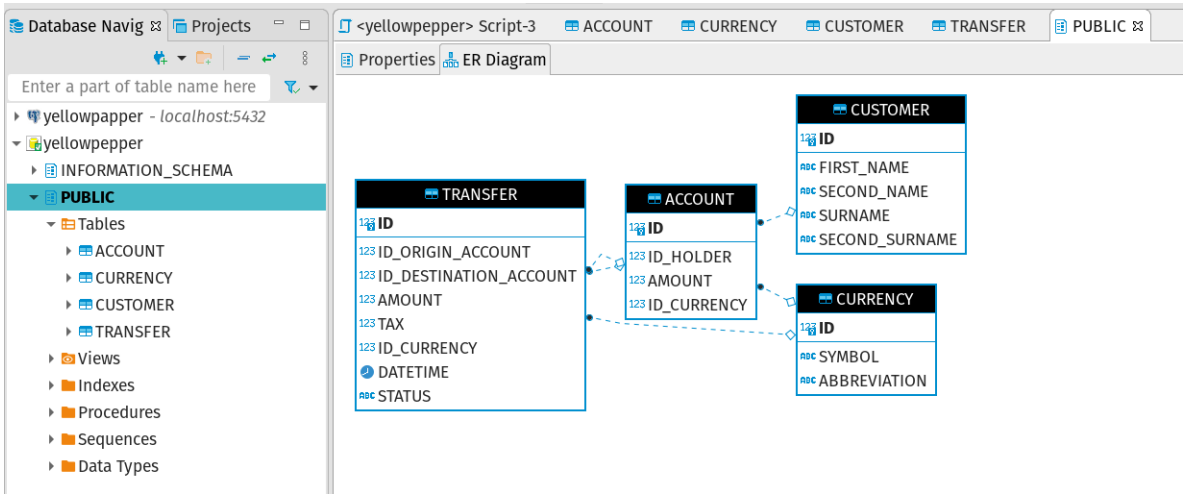

File Edit Navigate Search SQL Editor Database Window Help
Commit Rollback Auto yellowpapper public@y
Database Navig Projects
Enter a part of table name here
yellowpapper - localhost:5432
yellowpepper
Schemas
information_schema
pg_catalog
public
Tables
account
currency
customer
spatial_ref_sys
transfer
Views
Materialized Views
Indexes
Functions
Sequences
Data types
Aggregate functions
<yellowpepper> Script-3 transfer account currency
Properties Data ER Diagram
currency Enter a SQL expression to filter results (use Ctrl+Space)
Grid Text Record
id symbol abbreviation
1 1 $ USD
2 2 $ CAD
3 3 $ COP
4 4 € EUR
5 5 R$ BRL
Save Cancel Script
Project - General
#empty

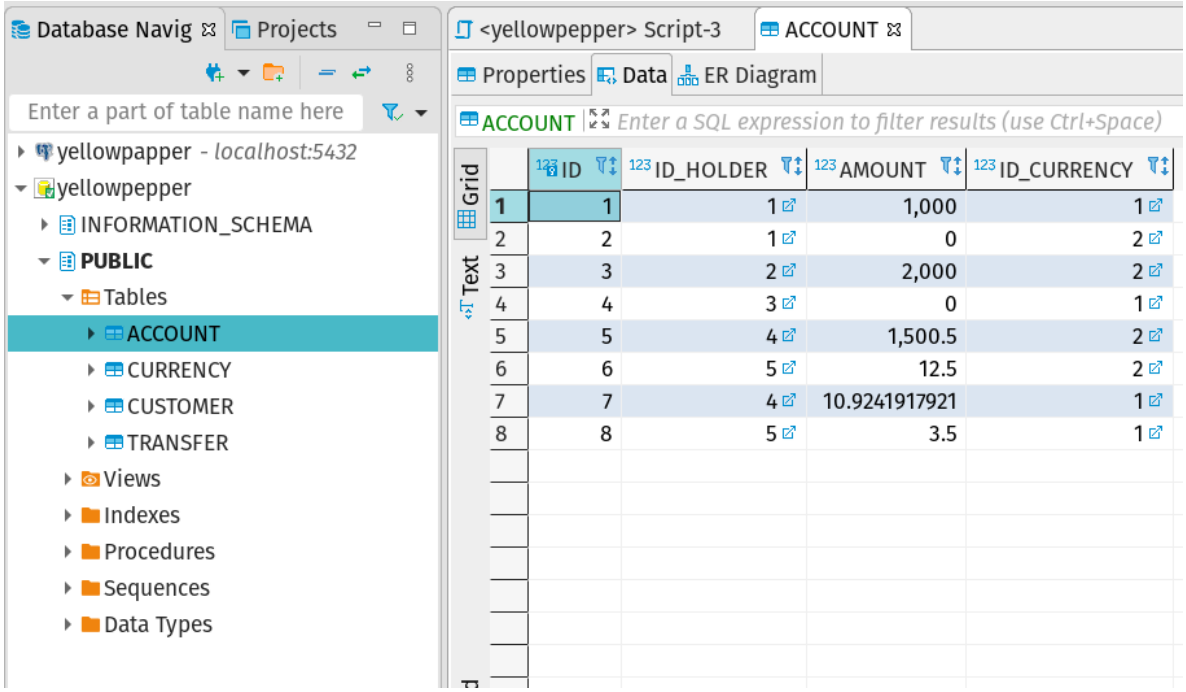

File Edit Navigate Search SQL Editor Database Window Help
Commit Rollback Auto yellowpapper public@yellowpepper
Database Navig Projects
Enter a part of table name here
yellowpapper - localhost:5432
yellowpepper
Schemas
information_schema
pg_catalog
public
Tables
account
currency
customer
spatial_ref_sys
transfer
Views
Materialized Views
Indexes
Functions
Sequences
Data types
Aggregate functions
<yellowpepper> Script-3 transfer account currency customer
Properties Data ER Diagram
customer Enter a SQL expression to filter results (use Ctrl+Space)
Grid Text Record
id first_name second_name surname second_surname
1 1 Jorge Ulises Useche Cuellar
2 2 John Doe
3 3 Jane Doe
4 4 Simón José Antonio de la Santísima Trinidad Bolívar y Ponte Palacios y Blanco
5 5 Jonny Deep
Save Cancel Script 200 5 Rows: 1
Project - General
#empty

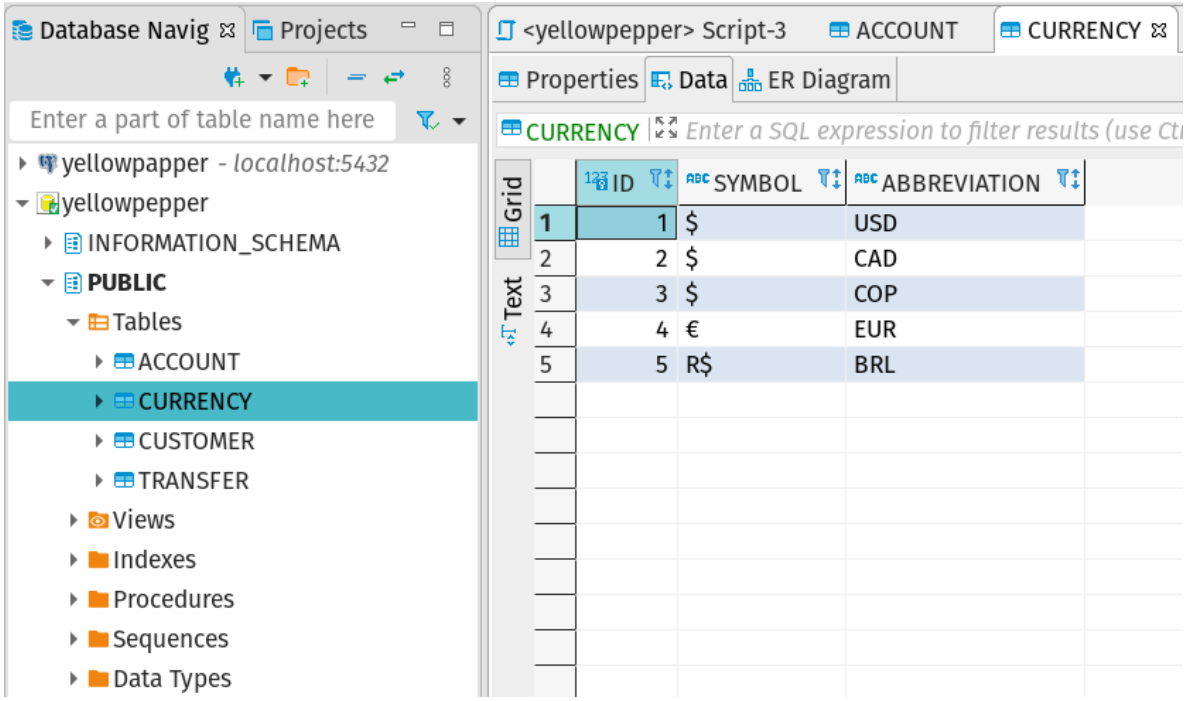


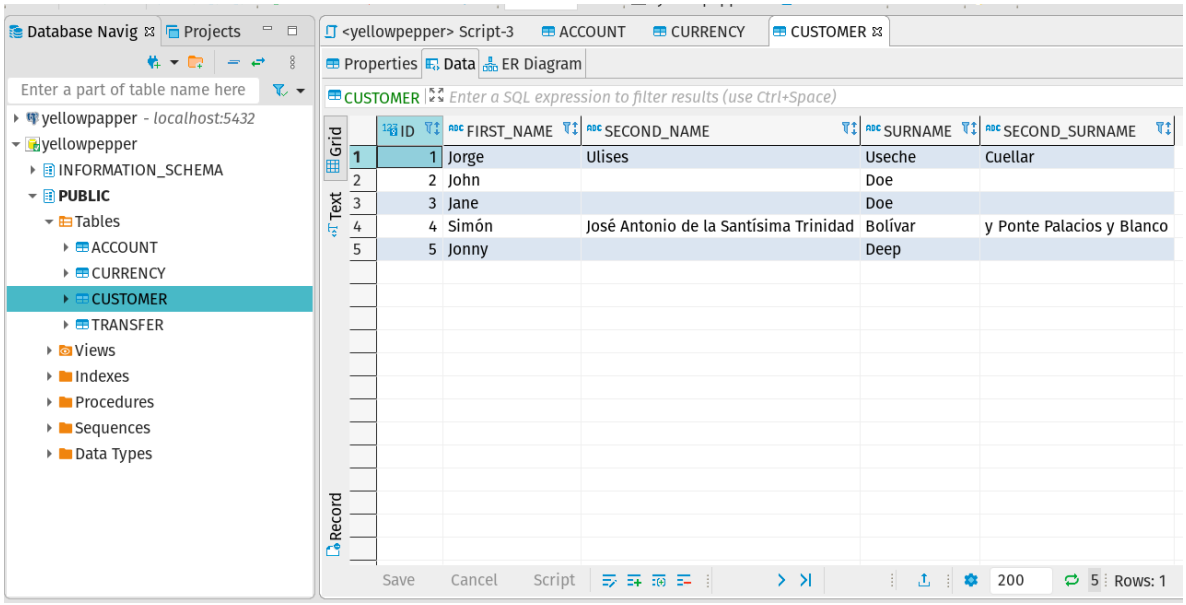

```
jorge@ulises:~/workspace/YellowPepper$ docker-compose up
Creating network "yellowpepper_docker-net" with driver "bridge"
Creating postgres ... done
Creating pgadmin4 ... done
Attaching to pgadmin4, postgres
postgres    | Add rule to pg_hba: 0.0.0.0/0
postgres    | Add rule to pg_hba: replication replicator
postgres    | Setup master database
postgres    | 2021-07-21 08:07:02.644 UTC [26] LOG:  starting PostgreSQL 12.2 (Debian 12.2-2.pg
postgres    | 2021-07-21 08:07:02.644 UTC [26] LOG:  listening on IPv4 address "127.0.0.1", por
postgres    | 2021-07-21 08:07:02.646 UTC [26] LOG:  listening on Unix socket "/var/run/postgre
postgres    | 2021-07-21 08:07:02.661 UTC [37] LOG:  database system was interrupted; last know
postgres    | 2021-07-21 08:07:02.661 UTC [38] postgres@postgres FATAL:  the database system is
postgres    | psql: error: could not connect to server: FATAL:  the database system is starting
postgres    | 2021-07-21 08:07:02.767 UTC [37] LOG:  database system was not properly shut down
postgres    | 2021-07-21 08:07:02.769 UTC [37] LOG:  redo starts at 0/42A107E8
```

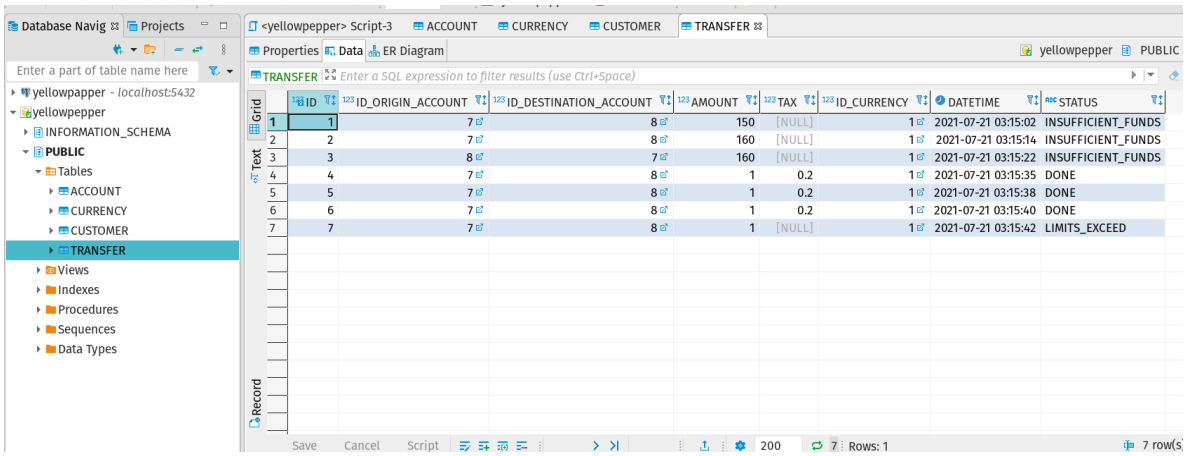
Data in h2 db file






Database Navig
Projects
Enter a part of table name here
yellowpapper - localhost:5432
yellowpepper
INFORMATION_SCHEMA
PUBLIC
Tables
ACCOUNT
CURRENCY
CUSTOMER
TRANSFER
Views
Indexes
Procedures
Sequences
Data Types
<yellowpepper> Script-3
ACCOUNT
CURRENCY
Properties
Data
ER Diagram
CURRENCY
Enter a SQL expression to filter results (use Ct
Grid
Text
ID
SYMBOL
ABBREVIATION
1 1 $ USD
2 2 $ CAD
3 3 $ COP
4 4 € EUR
5 5 R$ BRL


Database Navig
Projects
Enter a part of table name here
yellowpapper - localhost:5432
yellowpepper
INFORMATION_SCHEMA
PUBLIC
Tables
ACCOUNT
CURRENCY
CUSTOMER
TRANSFER
Views
Indexes
Procedures
Sequences
Data Types
<yellowpepper> Script-3
ACCOUNT
CURRENCY
CUSTOMER
Properties
Data
ER Diagram
CUSTOMER
Enter a SQL expression to filter results (use Ctrl+Space)
Grid
Text
Record
ID
FIRST_NAME
SECOND_NAME
SURNAME
SECOND_SURNAME
1 1 Jorge Ulises Useche Cuellar
2 2 John Doe
3 3 Jane Doe
4 4 Simón José Antonio de la Santísima Trinidad Bolívar y Ponte Palacios y Blanco
5 5 Jonny Deep
Save
Cancel
Script
200
5
Rows: 1

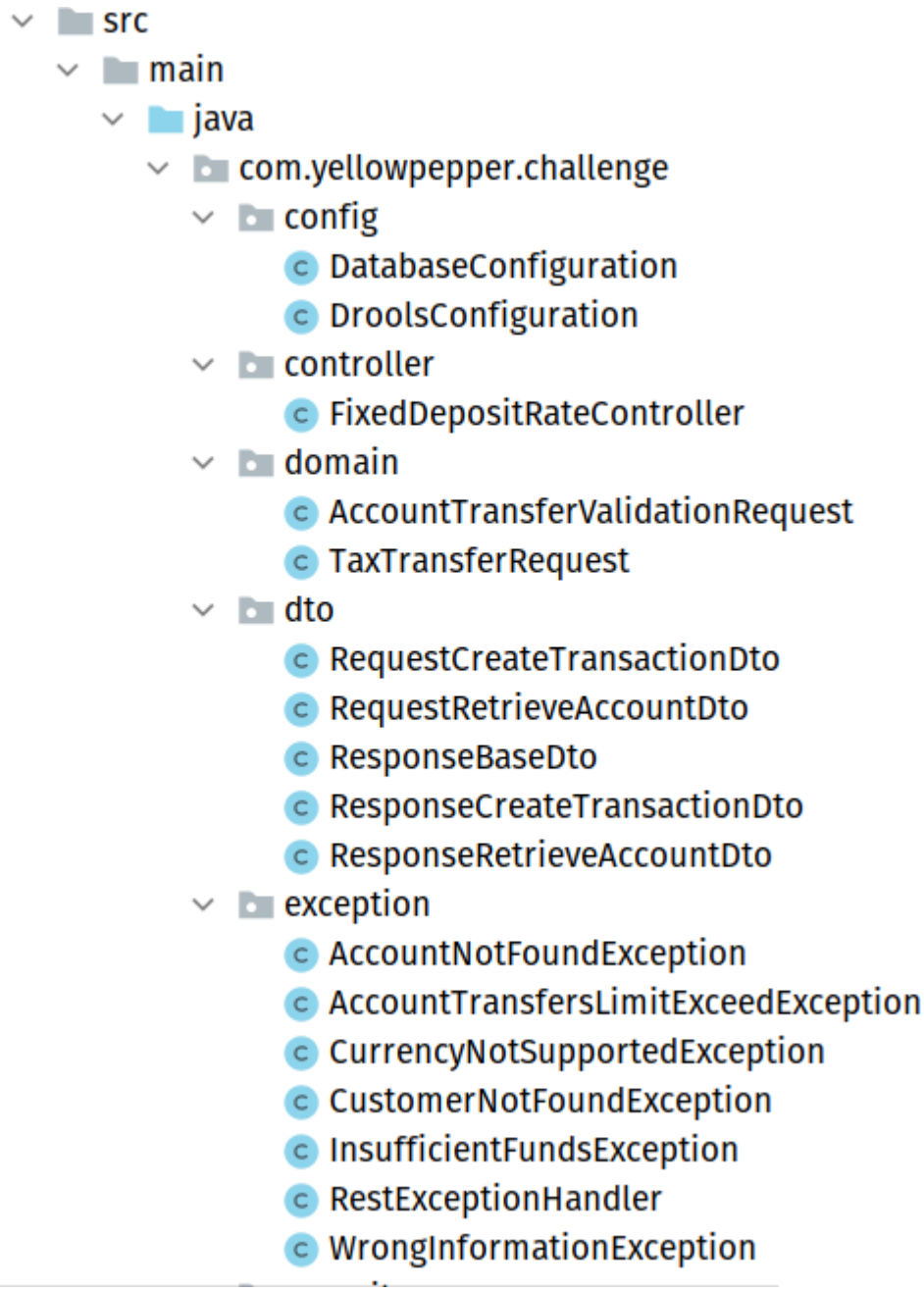


TDD with BDD focus

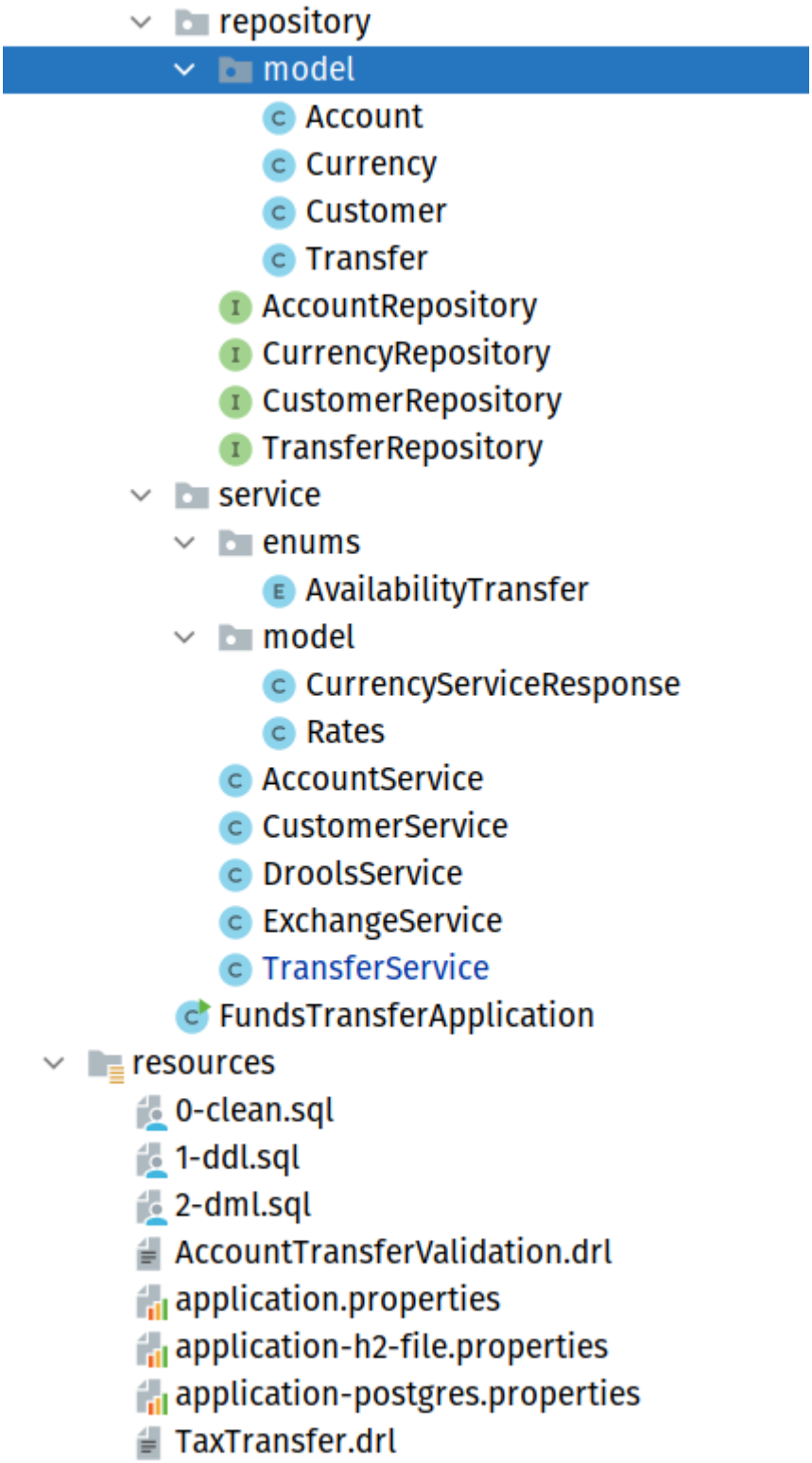
```
42  @Test
43  @DisplayName(
44      "Given valid accounts and valid parameters "
45          + "When user do a transaction with amount lower or equals to 100 "
46          + "Then the response is OK With 0.2 Tax ")
47  void test1() throws Exception {
48    ObjectNode transaction = getTransaction( amount: 100.0,  currency: "USD",  originAccount: 3,  destinationAccount: 4,  description: "Transferring across accounts");
49
50    HttpEntity<Object> entity = new HttpEntity<>(transaction);
51    ResponseEntity<ObjectNode> result =
52        testRestTemplate.exchange(TRANSACTIONS_ENDPOINT, HttpMethod.POST, entity, ObjectNode.class);
53    ObjectNode body = result.getBody();
54
55    assertEquals( expected: 200, result.getStatusCode().value());
56    assertEquals( expected: "OK", body.get("status").asText());
57    assertEquals( expected: 0, body.withArray( propertyName: "errors").size());
58    assertEquals( expected: 0.2, body.get("tax_collected").asDouble());
59    assertTrue(body.get("CAD").isDouble());
60  }
```

```java
@Test
@DisplayName(
    "Given valid accounts and valid parameters and sufficient amounts "
        + "When the user do a transaction 4 times "
        + "Then 4th time gives a limit error ")
void test4() throws Exception {
  ObjectNode transaction =
      getTransaction(
          amount: 1.0,  currency: "USD",  originAccount: 5,  destinationAccount: 6,  description: "Hey dude! I am sendi

  HttpEntity<Object> entity = new HttpEntity<>(transaction);
  ResponseEntity<ObjectNode> tx1 =
      testRestTemplate.exchange(TRANSACTIONS_ENDPOINT, HttpMethod.POST, entity, ObjectNode.class);
  assertEquals( expected: 200, tx1.getStatusCode().value());

  ResponseEntity<ObjectNode> tx2 =
      testRestTemplate.exchange(TRANSACTIONS_ENDPOINT, HttpMethod.POST, entity, ObjectNode.class);
  assertEquals( expected: 200, tx2.getStatusCode().value());

  ResponseEntity<ObjectNode> tx3 =
      testRestTemplate.exchange(TRANSACTIONS_ENDPOINT, HttpMethod.POST, entity, ObjectNode.class);
  assertEquals( expected: 200, tx3.getStatusCode().value());

  ResponseEntity<ObjectNode> tx4 =
      testRestTemplate.exchange(TRANSACTIONS_ENDPOINT, HttpMethod.POST, entity, ObjectNode.class);

  ObjectNode body = tx4.getBody();

  assertEquals( expected: 412, tx4.getStatusCode().value());
  assertEquals( expected: "KO", body.get("status").asText());
  assertEquals( expected: 1, body.withArray( propertyName: "errors").size());
```

Directory structure

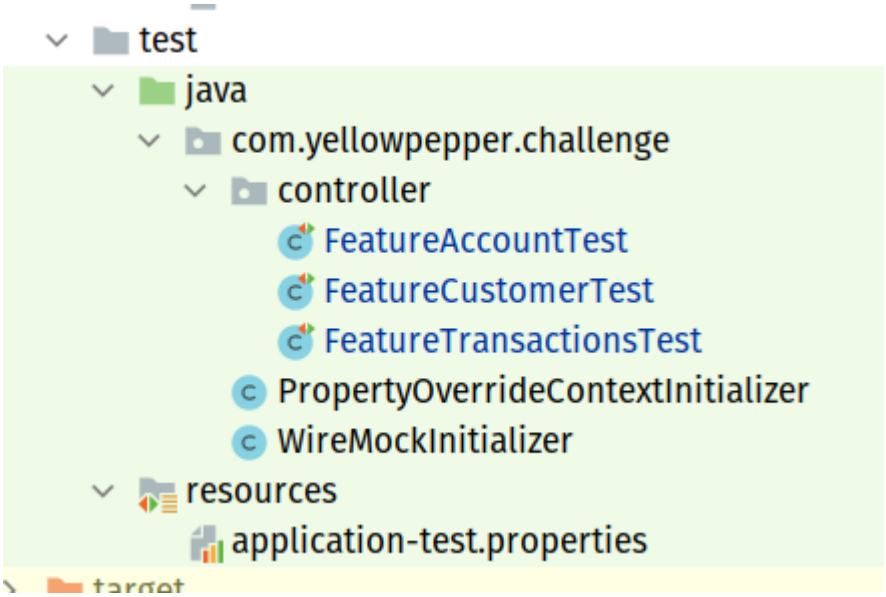


- Config: provides from general configuration at start of project like drools rules building and database population
- Controller: provides all the API resources with ther methods
- Domain: domain specific classes to read drools rules
- DTO: Objects of information transference
- Exception: Custom exceptions of the app



- Repository.model: the database models representations
- Repository: the repositories to access to database

- Service.enums: Enums to be used on services
- Service.model: Just some POJOS to transfer info
- Service: All the application services are here
- Resources: resources of the app like SQL scripts for population, configuration files to choose between different database, drools rules (DRL)



- Test: with the testing classes

Reactive programming

```
public Mono<ResponseCreateTransactionDto> applyDiscount(
    Transfer transfer, Account origin, Account destiny, Double tax, BigDecimal amountToTransfer) {
  Mono<Transfer> transferMono = transferRepository.save(transfer);

  return transferMono
      .flatMap(done -> convertCurrenciesToUSD(origin, destiny)) Mono<TransferService.NewAmounts>
      .map(
          oldAmountsInUSD -> {
            Double originInUSD = oldAmountsInUSD.getNewAmountOfOrigin();
            Double destinyInUSD = oldAmountsInUSD.getNewAmountOfDestiny();
            return getNewAmounts(tax, amountToTransfer.doubleValue(), originInUSD, destinyInUSD);
          })
      .flatMap(
          newAmountsInUSD ->
              convertCurrenciesToCAD(transfer, origin, destiny, newAmountsInUSD, tax)) Mono<Boolean>
      .flatMap(done -> exchangeService.fromCADtoUSD(1)) Mono<Double>
      .map(cadInUsd -> createResponse(BigDecimal.valueOf(tax), BigDecimal.valueOf(cadInUsd)));
}
```

```java
public Mono<Double> fromUSDtoCAD(double usd) {
  return getInfoCurrenciesFromService()
      .map(
          body -> {
            if (!body.getSuccess().booleanValue()) {
              throw new ResponseStatusException(
                  HttpStatus.SERVICE_UNAVAILABLE,
                  "CURRENCIES SERVICE FAILED CONNECTION NOT AVAILABLE TO TRANSFORM FROM USD TO CAD");
            } else {
              saveCache(body);
              double cad = usd * body.getRates().getCad();
              LOGGER.log(Level.INFO,  msg: "From USD to CAD: {0}USD", usd);
              LOGGER.log(Level.INFO,  msg: "From USD to CAD: {0}CAD", cad);
              return cad;
            }
          });
}
```