NEST HACKATHON

Submitted By

CARBON TEAM

ANOOP P M

JOHN CHRISTO

MANU FASIL M

Abstract

This Application is used for transfer money from one account to another accounts using mt103 parse method. When the user send a transfer request this application convert into MT103 Message Then Convert to MX message.

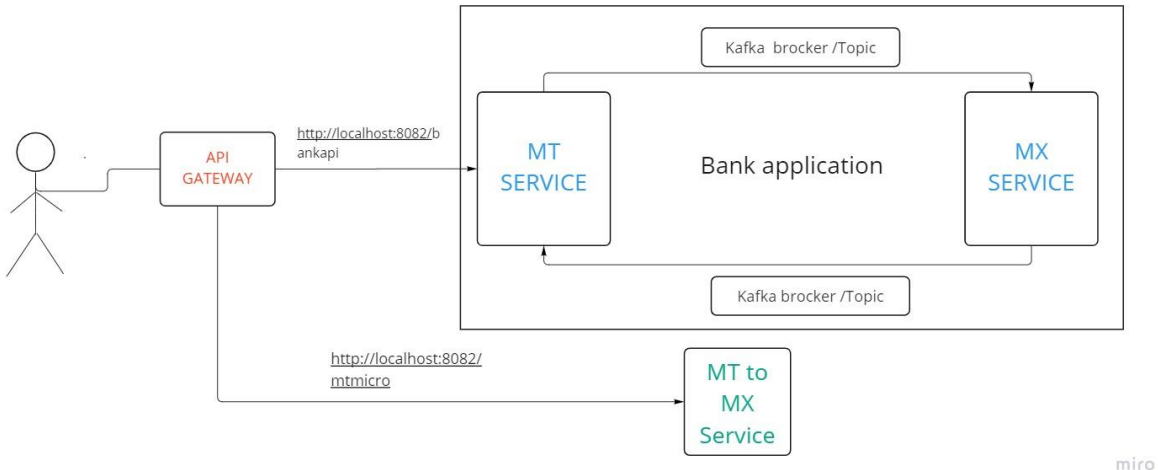
Used Technologies

- Java 8
- Spring boot
- Docker
- Kafka
- Mysql
- Swagger

Used Tools

- Docker Desktop
- Mysql Workbench
- Jmeter
- Spring tool
- Postman

ARCHITECTURE



Project Setup

- Install Java 8
- Set up Kafka Using Docker ,Docker compose file attached in project folder
- Create Kafka topic
  kafka-topics.sh --create --topic mtmessage --bootstrap-server localhost:9092

  kafka-topics.sh --create --topic mxmessage --bootstrap-server localhost:9092

  ## Listen Kafka Messages

  kafka-console-consumer.sh --topic mtmessage --from-beginning --bootstrap-server localhost:9092

{"SenderAccountnumber":"790773028412345","MTMessage":"{1:F01ENFEESS1A230000000000}{2:I103ENF43332XXXXN}{3:{121:724209c7-6d6e-4a68-becd-6c51bb0997bf}}{4:\r\n:20:REFERENCE\r\n:23B:CRED\r\n:32A:220829INR1\r\n:50A:/790773028412345\r\nSBI\r\n:59:/9539931867123\r\nSBI\r\n:71A:OUR\r\n-}","time":"2022-08-29T09:50:12.369307200Z"}

  kafka-console-consumer.sh --topic mxmessage --from-beginning --bootstrap-server localhost:9092

{"MXmessage":"<message>\r\n<block1>\r\n\t<applicationId>F<\/applicationId>\r\n\t<serviceId>01<\/serviceId>\r\n\t<logicalTerminal>0000000000<\/logicalTerminal>\r\n<\/block1>\r\n<block2 type=\"input\">\r\n\t<messageType>103<\/messageType>\r\n\t<receiverAddress>N<\/receiverAddress>\r\n<\/block2>\r\n<block3>\r\n\t<tag>\r\n\t\t<name>121<\/name>\r\n\t\t<value>6e7e8273-8467-4b0b-8b3a-0bc9a1d356a5<\/value>\r\n\t<\/tag>\r\n<\/block3>\r\n<block4>\r\n\t<tag>\r\n\t\t<name>20<\/name>\r\n\t\t<value>REFERENCE<\/value>\r\n\t<\/tag>\r\n\t<tag>\r\n\t\t<name>23B<\/name>\r\n\t\t<value>CRED<\/value>\r\n\t<\/tag>\r\n\t<tag>\r\n\t\t<name>32A<\/name>\r\n\t\t<value>220829100<\/value>\r\n\t<\/tag>\r\n\t<tag>\r\n\t\t<name>50A<\/name>\r\n\t\t<value>/17203647899955554\r\nSBI<\/value>\r\n\t<\/tag>\r\n\t<tag>\r\n\t\t<name>59<\/name>\r\n\t\t<value>/11225546165621\r\nSBI<\/value>\r\n\t<\/tag>\r\n\t<tag>\r\n\t\t<name>71A<\/name>\r\n\t\t<value>OUR<\/value>\r\n\t<\/tag>\r\n<\/block4>\r\n<\/message>"}

- Install mysql and give password as root (password = root) and create Database mtbank(DB = mtbank)
- Then Run the 4 Spring boot applications using spring or other tools

Hackathon API Request Examples

Make sure all four API are running successfully

1. Create User

   URI = http://localhost:8082/bankapi/usercreate (post method)

   Json Request body =

   ```
   {
     "accountnumber": "1720364789995555",
     "username": "Ajay"
   }
   ```

2. Create Receivers bank details

   URI = http://localhost:8082/bankapi/addreceiveraccount (post method)
   Json Request body =

   ```
   {
     "accountnumber": "11225546165620",
     "bankName": "SBI",
     "ifsccode": "SBIN112445",
     "receivername": "Arshad"
   }
   ```

3. User Deposit

   URI = http://localhost:8082/bankapi/deposit (put method)
   Json Request body =

   ```
   {
     "accountnumber": "1720364789995555"
   }
   ```

4. Transfer amount :this api take user data and create mt103 then send the mt103 message

   URI = http://localhost:8082/bankapi/transfermessage (post method)

   Json Request body =

{

"accountnumber": "790773028412345",
"address": "SBI",
"amount": "100",
"bankname": "SBI",
"currency": "INR",
"receiver": "ENF43332",
"receiverAccountNo": "9539931867123",
"refernce": "CRED",
"sender": "ENFEESS123 }

5. User Balance

URI =http://localhost:8082/bankapi/userbalance (Get Method)

Json request body:
{
"accountnumber": "1720364789995555"
}

6. Receiver Balance

URI =http://localhost:8082/bankapi/receiverbalance (Get Method)

Json request body:
{
"accountnumber": "1244"
}

7. MT to MX Converter

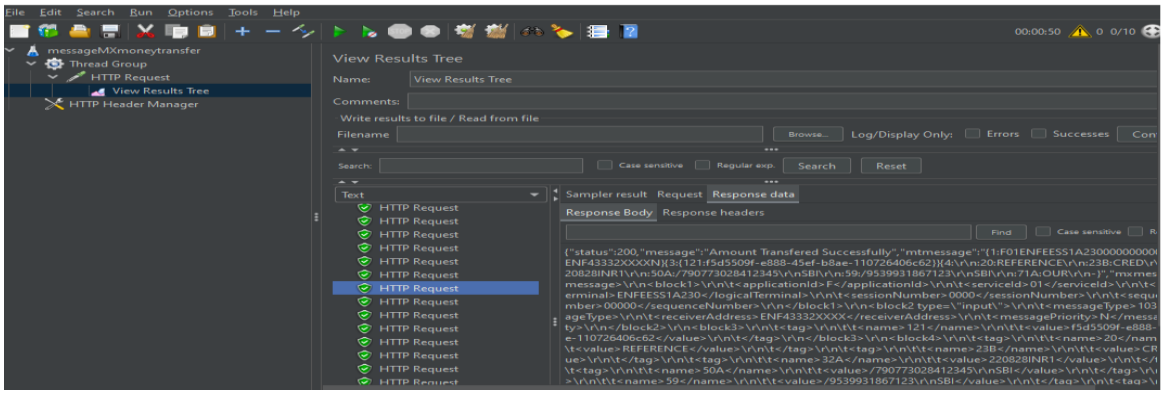
URI = http://localhost:8082/mtmicro/mttovalue (Get Method)

Json request body: This message takes from given 103.txt file

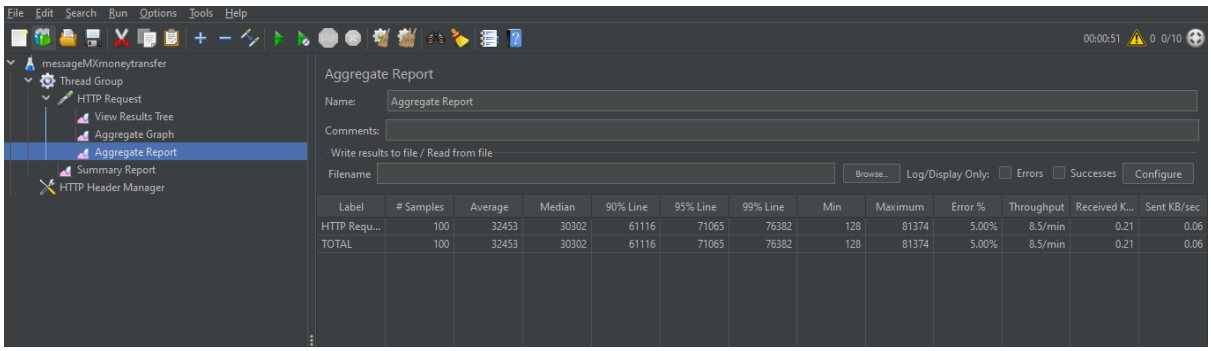

```
{
    "message":"{1:F01BICFOOYYAXXX8683497519}{2:O1031535051028ESPBESMMAXXX5423752247
0510281535N}{3:{113:ROMF}{108:0510280182794665}{119:STP}}{4:\r\n:20:006135011308990
8\r\n:13C:/RNCTIME/1534+0000\r\n:23B:CRED\r\n:23E:SDVA\r\n:32A:061028EUR100000,\r\n
:33A:081029EUR120000,\r\n:33B:EUR100000,\r\n:50K:/12345678\r\nAGENTES DE BOLSA FOO
AGENCIA\r\nAV XXXXX 123 BIS 9 PL\r\n12345 BARCELONA\r\n:52A:/2337\r\nFOOAESMMXXX\r\
n:53A:FOOAESMMXXX\r\n:57A:BICFOOYYXXX\r\n:59:/ES0123456789012345671234\r\nFOO AGENT
ES DE BOLSA ASOC\r\n:71A:OUR\r\n:72:/BNF/TRANSF. BCO. FOO\r\n-
}{5:{MAC:88B4F929}{CHK:22EF370A4073}}"
}
```

Project Performance Test

Using JMETER

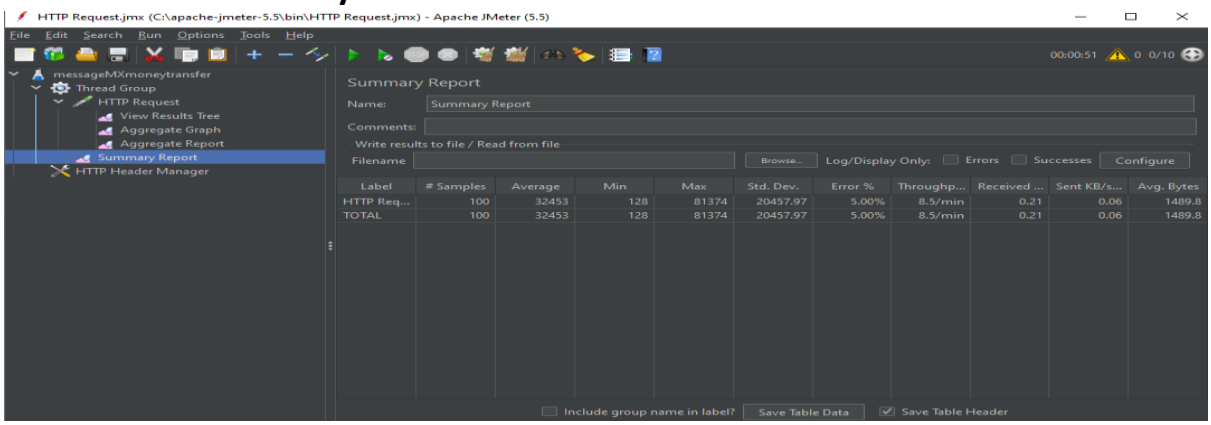


AGGREGATE REPORT

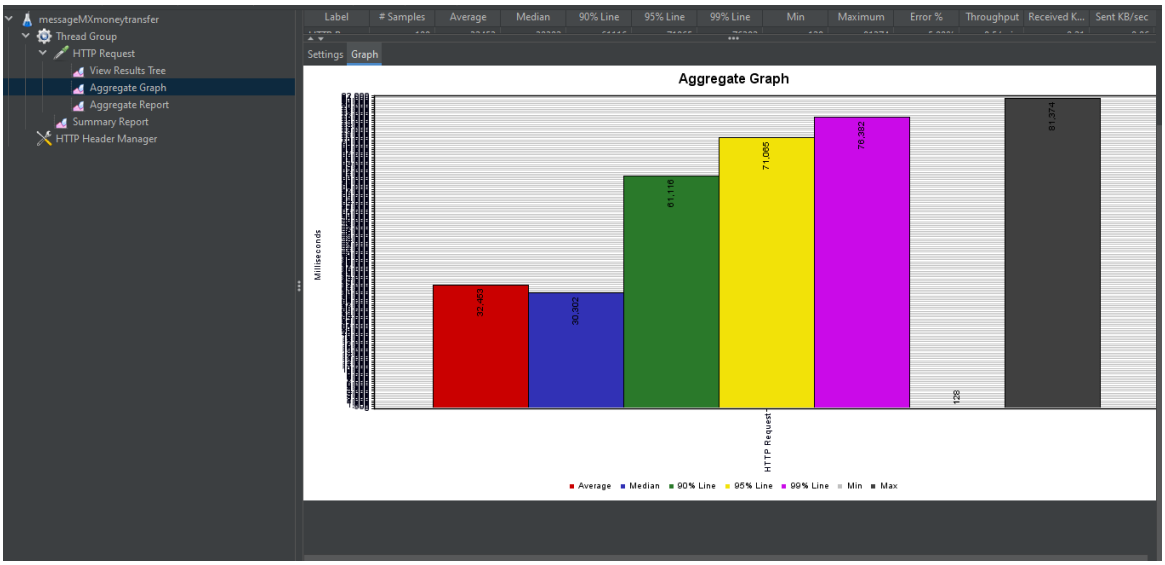


Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/...	Sent KB/sec	Avg. Bytes
HTTP Request	10	25117	5150	45104	12747.95	0.00%	11.7/min	0.30	0.08	1557.0
TOTAL	10	25117	5150	45104	12747.95	0.00%	11.7/min	0.30	0.08	1557.0

AGGREGATE Summary

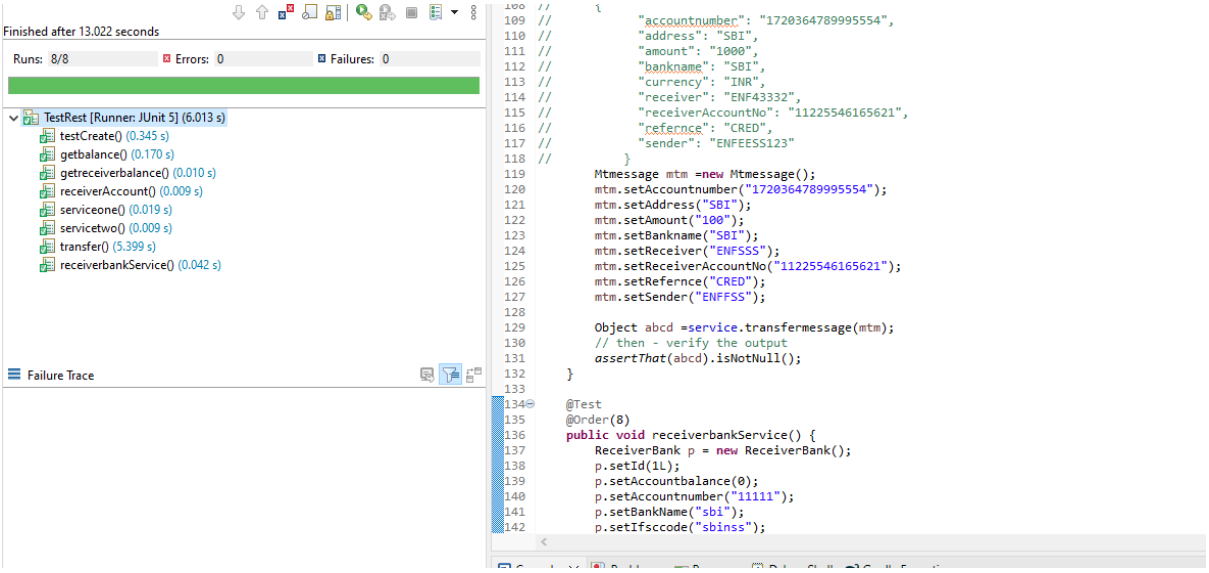


Graph



JUNIT TESTING

Junit test for 7 modules



SWAGGER IMPLIMENTATION