

# NEST HACKATHON

ARCHITECTURE AND DESIGN DOCUMENT

**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.

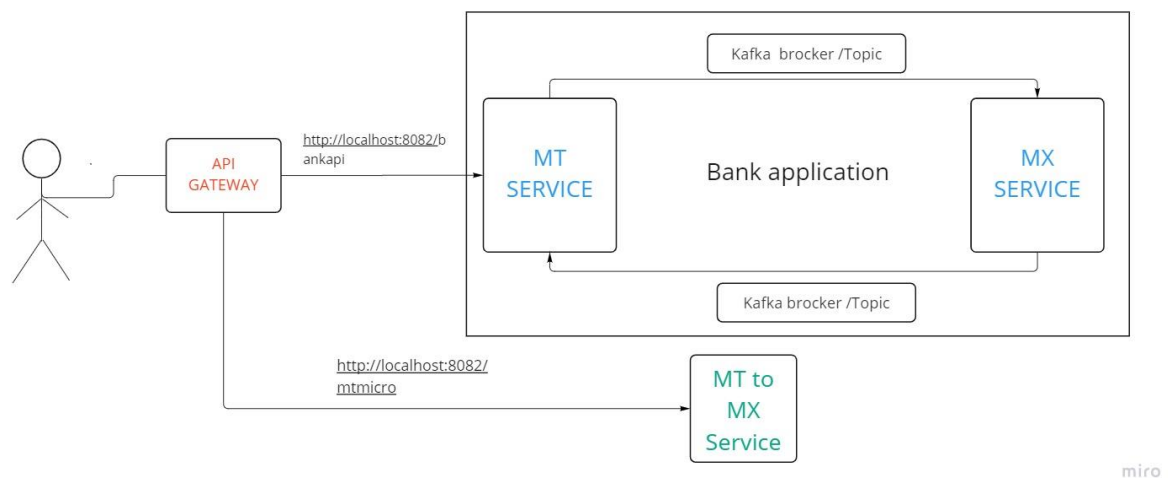
### Used Technologies

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

### Used Tools

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

## ARCHITECTURE AND DESIGN



## API GATEWAY

This Service is provides a flexible way of routing requests based on a number of criteria, as well as focuses on cross-cutting concerns such as security, resiliency, and monitoring. Spring Cloud Gateway aims to provide a simple, yet effective way to route to APIs and provide cross cutting concerns to them such as: security, monitoring/metrics, and resiliency.

Here we used to route the 2 micro services.

## MT Service

This Service is used to convert User request to MT103 messages then This application produce message to a kafka topics(mtmessage).Also listening the kafka topic and Consume the messages and give a response to user .

- Also provide user registration for creating user account.
- Store Receiver account details.
- User Deposit
- Check user balance .

When we create account it is a zero balance account . All transactions are reflected through this given datas.

## MX Service

This service is used to convert MT103 message to MX message .Using kafka consume technology for listening messages then this converted XML message produce to a kafka topics(mx message).

## MT to MX Service .

This service is used to convert MT103 message to MX message Directly .Then give a response to user

## 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: 103ENF43332XXXXN",
    "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\r\n\t<serviceld>01</serviceld>\r\n\r\n\tlogicalTerminal>0000000000</logicalTerminal>\r\n\r\n\t</block1>\r\n\r\n\t<block2  
type=\"input\">\r\n\r\n\t<messageType>103</messageType>\r\n\r\n\t<receiverAddress>N</receiverAddress>\r\n\r\n\t</block2>\r\n\r\n\t<blo  
ck3>\r\n\r\n\t<tag>\r\n\r\n\t\t<tname>12I</tname>\r\n\r\n\t\t<tvalue>6e7e8273-8467-4b0b-8b3a-  
0bc9a1d356a5</tvalue>\r\n\r\n\t\t</tag>\r\n\r\n\t</block3>\r\n\r\n\t<block4>\r\n\r\n\t\t<tag>\r\n\r\n\t\t\t<tname>20</tname>\r\n\r\n\t\t\t<tvalue>REFEREN  
CE</tvalue>\r\n\r\n\t\t\t</tag>\r\n\r\n\t\t\t<tag>\r\n\r\n\t\t\t\t<tname>23B</tname>\r\n\r\n\t\t\t\t<tvalue>CRED</tvalue>\r\n\r\n\t\t\t\t</tag>\r\n\r\n\t\t\t\t<tag>\r\n\r\n\t\t\t\t\t<tname>  
32A</tname>\r\n\r\n\t\t\t\t\t<tvalue>220829100</tvalue>\r\n\r\n\t\t\t\t\t</tag>\r\n\r\n\t\t\t\t\t<tag>\r\n\r\n\t\t\t\t\t\t<tname>50A</tname>\r\n\r\n\t\t\t\t\t\t<tvalue>/1  
720364789995554\r\n\r\n\tSBI</tvalue>\r\n\r\n\t\t\t\t\t\t</tag>\r\n\r\n\t\t\t\t\t\t<tag>\r\n\r\n\t\t\t\t\t\t\t<tname>59</tname>\r\n\r\n\t\t\t\t\t\t\t<tvalue>/11225546165621\r\n\r\n\tSBI  
</tvalue>\r\n\r\n\t\t\t\t\t\t\t</tag>\r\n\r\n\t\t\t\t\t\t\t<tag>\r\n\r\n\t\t\t\t\t\t\t\t<tname>71A</tname>\r\n\r\n\t\t\t\t\t\t\t\t<tvalue>OUR</tvalue>\r\n\r\n\t\t\t\t\t\t\t\t</tag>\r\n\r\n\t\t\t\t\t\t\t</block4>\r\n\r\n\t</  
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",
  "ifsc": "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/transferrmessage> (post method)

Json Request body =

```
{
  "accountnumber": "790773028412345",
  "address": "SBI",
  "amount": "100",
  "bankname": "SBI",
  "currency": "INR",
  "receiver": "ENF43332",
  "receiverAccountNo": "9539931867123",
  "reference": "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}{2:01031535051028ESPBESMMAXX5423752247  
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}}"  
}
```

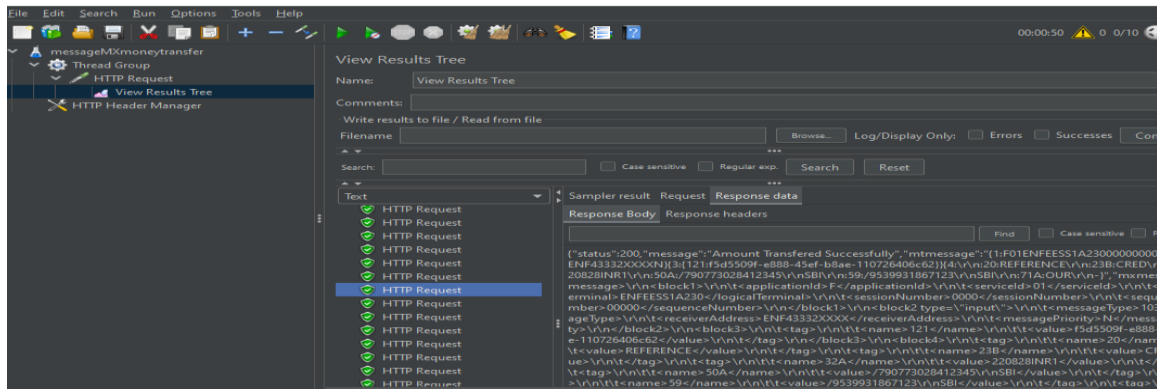
# Project Performance Test

## Using JMETER

Here we give 2 threads and 100 loops in one second ramp period so we get zero Error%.

When we give 10 thread with 50 loops the error percentage is 5% .

Here only showing our major rest api call for Transfer money .



## AGGREGATE REPORT

Aggregate Report													
Name: Aggregate Report													
Comments:													
Write results to file / Read from file													
Filename: Browse...													
Log/Display Only: Errors Successes Configure													
Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received KB...	Sent KB/sec	
HTTP Reque...	210	11127	10279	10304	10315	41308	5157	51395	0.00%	11.3/min	0.29	0.08	
TOTAL	210	11127	10279	10304	10315	41308	5157	51395	0.00%	11.3/min	0.29	0.08	

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

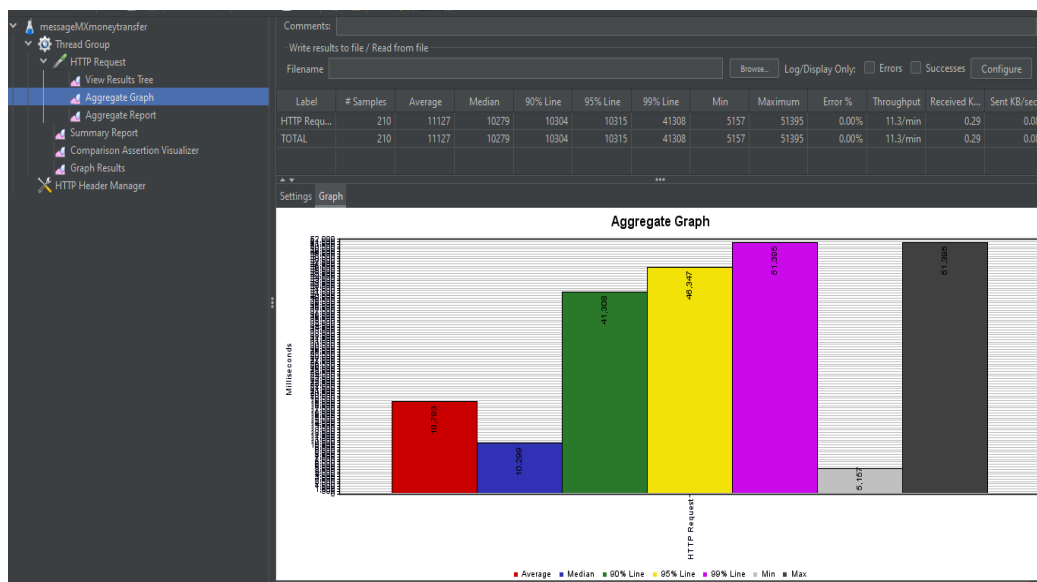
## AGGREGATE Summary

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/...	Sent KB/sec	Avg. Bytes
HTTP Request	210	11127	5157	51395	5055.50	0.00%	11.3/min	0.29	0.08	1551.9
TOTAL	210	11127	5157	51395	5055.50	0.00%	11.3/min	0.29	0.08	1551.9

## AGGREGATE Summary

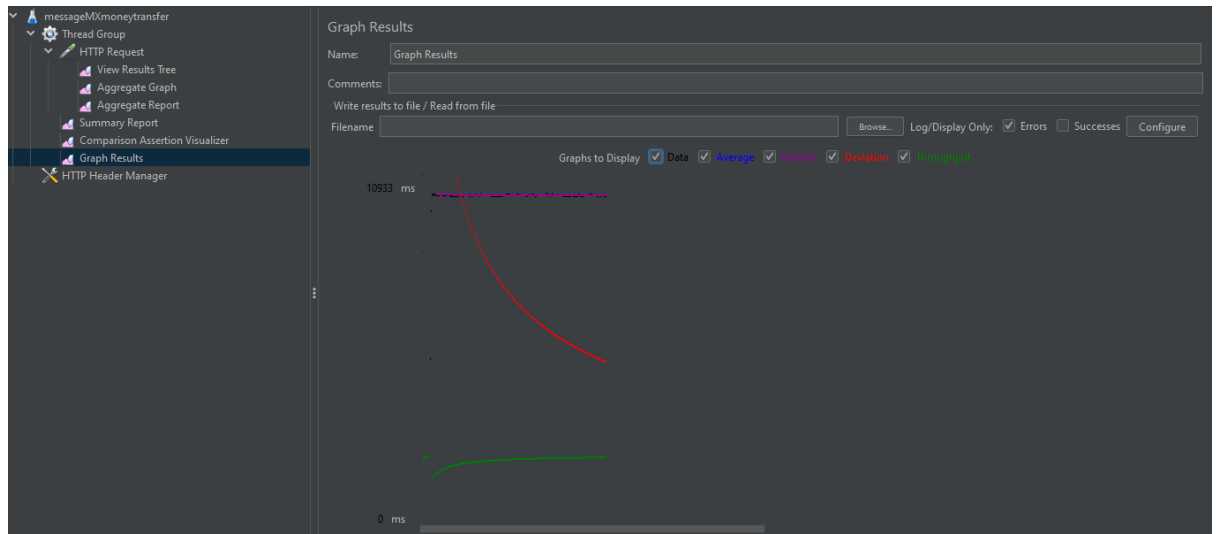
Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received ...	Sent KB/s...	Avg. Bytes
HTTP Request	100	32433	128	81374	20437.87	5.00%	8.5/min	0.21	0.06	1489.3
TOTAL	100	32433	128	81374	20437.87	5.00%	8.5/min	0.21	0.06	1489.3

## AGGREGATE Graph





## GRAPH



## JUNIT TESTING

JUnit test for 7 modules

Here we created 8 Unit test cases for all rest api services . All service tests are passed .

```
Finished after 13.022 seconds
Runs: 8/8 Errors: 0 Failures: 0

TestRest [Runner: JUnit 5] (6.013 s)
  testCreate() (0.345 s)
  getBalance() (0.170 s)
  getReceiverBalance() (0.010 s)
  receiverAccount() (0.009 s)
  serviceOne() (0.019 s)
  serviceTwo() (0.009 s)
  transfer() (5.399 s)
  receiverBankService() (0.042 s)

Failure Trace

1100 //      {
1101 //          "accountnumber": "1720364789995554",
1102 //          "address": "SBI",
1103 //          "amount": "1000",
1104 //          "bankname": "SBI",
1105 //          "currency": "INR",
1106 //          "receiver": "ENF43332",
1107 //          "receiverAccountNo": "11225546165621",
1108 //          "reference": "CRED",
1109 //          "sender": "ENFEESS123"
1110 //      }
1111 //
1112 //      Mtmmessage mtm =new Mtmmessage();
1113 //      mtm.setAccountnumber("1720364789995554");
1114 //      mtm.setAddress("SBI");
1115 //      mtm.setAmount("1000");
1116 //      mtm.setBankname("SBI");
1117 //      mtm.setReceiver("ENFSSS");
1118 //      mtm.setReceiverAccountNo("11225546165621");
1119 //      mtm.setReference("CRED");
1120 //      mtm.setSender("ENFESS");
1121 //
1122 //      Object abcd =service.transfermessage(mtm);
1123 //      // then - verify the output
1124 //      assertThat(abcd).isNotNull();
1125 //  }
1126 //
1127 //  @Test
1128 //  @Order(8)
1129 //  public void receiverBankService() {
1130 //      ReceiverBank p = new ReceiverBank();
1131 //      p.setId(1L);
1132 //      p.setAccountbalance(0);
1133 //      p.setAccountnumber("11111");
1134 //      p.setBankName("sbi");
1135 //      p.setIfscCode("sbinss");
1136 //  }
1137 //  }
```

## SWAGGER IMPLIMENTATION

We use the swagger for the rest control testing and documentation.

# Bank Solution <sup>1.0.0</sup>

[ Base URL: localhost:8080 / ]  
<http://localhost:8080/v2/api-docs>

"Transfer money from one Account to Another Account"

[Anoop P.M - Website](#)  
[Send email to Anoop P.M](#)  
[Apache License Version 2.0](#)

## message-controller Message Controller

POST	/bankapi/addreceiveraccount	Register A Receiver User
PUT	/bankapi/deposit	Deposit to user account
GET	/bankapi/receiverbalance	Register A User
POST	/bankapi/transfermessage	Transfer amount
GET	/bankapi/userbalance	Check user account balance
POST	/bankapi/usercreate	Register A User