Code Challenge

The goal of this code challenge is to create a microservice using Java and any framework that you think it is appropriate.

When you have it ready please upload the code to Github (on a **private** repository) and share it in read-only mode with user 'codechallengebackend'. After that, send an email to codechallenge.backend@orangebank.es to notify us.

It is VERY IMPORTANT to follow these rules:

- The repository in Github must be private
- No reference to Orange or Orangebank must be present anywhere in the code (repo name, packages, classes...)

Not following those rules will automatically discard the test and the candidate

Also, please add a file named 'README.md' detailing how to run/test the code and anything else that you'd like to share with us.

General rules

- Apply SOLID principles
- Do ATDD
- The requirements have been kept simple on purpose but structure and code the solution as if it were a big application
- The microservice should be auto-contained. Don't depend on any external services being run (ex. if needed, use an in-memory data storage)
- Some requirements have been left ambiguous on purpose so, if you make any assumption, please add a comment

Requirements

You have to build a microservice that will handle bank transactions. In order to do that you'll need to create the following endpoints.

Create transaction

This endpoint will receive the transaction information and store it into the system.

It is IMPORTANT to note that a transaction that leaves the total account balance bellow 0 is not allowed.

Payload:

```
{
  "reference":"12345A"
  "account_iban":"ES9820385778983000760236",
  "date":"2019-07-16T16:55:42.000Z",
  "amount":193.38,
  "fee":3.18,
  "description":"Restaurant payment"
}
```

- reference (optional): The transaction unique reference number in our system. If not present, the system will generate one.
- account_iban (mandatory): The IBAN number of the account where the transaction has happened.
- date (optional): Date when the transaction took place
- amount (mandatory): If positive the transaction is a credit (add money) to the account. If negative it is a debit (deduct money from the account)
- fee (optional): Fee that will be deducted from the amount, regardless on the amount being positive or negative.
- description (optional): The description of the transaction

Search transactions

This endpoint searches for transactions and should be able to:

- Filter by account_iban
- Sort by amount (ascending/descending)

Transaction status

This endpoint, based on the payload and some business rules, will return the status and additional information for a specific transaction.

Payload:

```
{
  "reference":"12345A",
  "channel":"CLIENT"
}
```

- reference (mandatory): The transaction reference number
- channel (optional): The type of the channel that is asking for the status. It can be any of these values: CLIENT, ATM, INTERNAL

Response:

```
{
  "reference":"12345A",
  "status":"PENDING",
  "amount":193.38,
  "fee":3.18
}
```

- reference: The transaction reference number
- status: The status of the transaction. It can be any of these values: PENDING, SETTLED, FUTURE, INVALID
- amount: the amount of the transaction
- fee: The fee applied to the transaction

Business Rules

A)

```
 Given: A transaction that is not stored in our system
When: I check the status from any channel
Then: The system returns the status 'INVALID'
```

Example payload:

```
{
  "reference":"XXXXXX",
  "channel":"CLIENT"
}
```

Example response:

```
{
  "reference":"XXXXXX",
  "status":"INVALID"
}
```

B)

```
 Given: A transaction that is stored in our system
When: I check the status from CLIENT or ATM channel
      And the transaction date is before today
Then: The system returns the status 'SETTLED'
      And the amount substracting the fee
```

Example payload:

```
{
  "reference":"12345A",
  "channel":"CLIENT"
}
```

Example response:

```
{
  "reference":"12345A",
  "status":"SETTLED",
  "amount":190.20
}
```

C)

```
 Given: A transaction that is stored in our system
When: I check the status from INTERNAL channel
      And the transaction date is before today
Then: The system returns the status 'SETTLED'
      And the amount
      And the fee
```

Example payload:

```
{
  "reference":"12345A",
  "channel":"INTERNAL"
}
```

Example response:

```
{
  "reference":"12345A",
  "status":"SETTLED",
  "amount":193.38,
  "fee":3.18
}
```

D)

```
 Given: A transaction that is stored in our system
When: I check the status from CLIENT or ATM channel
      And the transaction date is equals to today
Then: The system returns the status 'PENDING'
      And the amount substracting the fee
```

Example payload:

```
{
  "reference":"12345A",
  "channel":"ATM"
}
```

Example response:

```
{
  "reference":"12345A",
  "status":"PENDING",
  "amount":190.20
}
```

E)

```
 Given: A transaction that is stored in our system
When: I check the status from INTERNAL channel
      And the transaction date is equals to today
Then: The system returns the status 'PENDING'
      And the amount
      And the fee
```

Example payload:

```
{
  "reference":"12345A",
  "channel":"INTERNAL"
}
```

Example response:

```
{
  "reference":"12345A",
  "status":"PENDING",
  "amount":193.38,
  "fee":3.18
}
```

F)

```
 Given: A transaction that is stored in our system
When: I check the status from CLIENT channel
      And the transaction date is greater than today
Then: The system returns the status 'FUTURE'
      And the amount substracting the fee
```

Example payload:

```
{
  "reference":"12345A",
  "channel":"CLIENT"
}
```

Example response:

```
{
  "reference":"12345A",
  "status":"FUTURE",
  "amount":190.20
}
```

G)

```
 Given: A transaction that is stored in our system
When: I check the status from ATM channel
      And the transaction date is greater than today
Then: The system returns the status 'PENDING'
      And the amount substracting the fee
```

Example payload:

```
{
  "reference":"12345A",
  "channel":"ATM"
}
```

Example response:

```
{
  "reference":"12345A",
  "status":"PENDING",
  "amount":190.20
}
```

H)

```
 Given: A transaction that is stored in our system
When: I check the status from INTERNAL channel
      And the transaction date is greater than today
Then: The system returns the status 'FUTURE'
      And the amount
      And the fee
```

Example payload:

```
{
  "reference":"12345A",
  "channel":"INTERNAL"
}
```

Example response:

```
{
  "reference":"12345A",
  "status":"FUTURE",
  "amount":193.38,
  "fee":3.18
}
```