Hello candidate!

Nice to have you in this test practice step!

In this first step you will develop a simple service with 3 endpoints so that we can proceed to the next step.

The next step will take place after you submit the first step and when your interviewer schedules it. We will develop together (with one Pismo developer and you) a predefined solution on top of your previously created service that must take at most 1:30 hours to complete.

For this day, you must bring a small part of the solution already done and we will only add new features to the application.
In addition, this step will occur remotely and we will need that you share your screen with us, ok?

In the next page we will describe what must be done for the scheduled date.

Important technical notes:

- You can develop the solution using any language, but preferentially we recommend: **Java**, **Groovy, Kotlin** or **Go** ;
- The solution must be published at github and must contain a README with the necessary instructions to run the application.

Rating criteria:

1. Manuntenlability;
2. Simplicity;
3. Testability;
4. Documentation;

Bonus:

- We like docker;
- Easy application execution (./run script);
- Good documentation makes life easier.
- Tests are your friend!

If any questions arise feel free to question, call, make contact or send smoke signals - we are at your service.

pismo

Cheers,
Team Pismo

Transactions routine

Each cardholder (customer) has an account with their data.
For each operation done by the customer a transaction is created and associated with their respective account.
Each transaction has a specific type (normal purchase, withdrawal, credit voucher or purchase with installments)

Transactions of type purchase and withdrawal are registered with negative amounts, while transactions of credit voucher are registered with positive value.

Data structure

The following diagram shows a **suggested** structure for this test. **(Feel free to create your own model!)**

Accounts

Account_ID	**Document_Number**
1	12345678900

OperationsTypes

OperationType_ID	**Description0**
1	Normal Purchase
2	Purchase with installments
3	Withdrawal
4	Credit Voucher

Transactions

Transaction_ID	**Account_ID**	**OperationType_ID**	**Amount**	**EventDate**
1	1	1	-50.0	2020-01-01T10:32:07.7199222
2	1	1	-23.5	2020-01-01T10:48:12.2135875

3	1	1	-18.7	2020-01-02T19:01:23.1458543
4	1	4	60.0	2020-01-05T09:34:18.5893223

In table **Transactions** , the column **Amount** holds the value of the transaction and the column **EventDate** holds the moment that the transaction occurred.

Endpoints

Create the endpoints below according to the use cases previously mentioned.

```
POST /accounts (create an account)
Request Body:
{
	"document_number": "12345678900"
}

GET /accounts/:accountId (retrieve the account information)
Response Body:
{
	"account_id": 1,
	"document_number": "12345678900"
}

POST /transactions (create a transaction)
Request Body:
{
	"account_id": 1,
	"operation_type_id": 4,
	"amount": 123.45
}
```