Assignment

Background

As a member of a small team that has been tasked with quickly assembling a prototype of a small web application for an upcoming customer demo, you were made responsible for designing and implementing its back-end part.

Meant as home budgeting aid, the planned system will need to expose multiple endpoints to support all desired functionalities. For now, however, it has been decided that only 3 endpoints and data persistence need to be operational for the upcoming demo.

The prototype

Planned home budgeting assistant is meant to operate on so called registers. A register is conceptually similar to a bank account – one can transfer money to and from it, and both of these operations directly affect its balance. The system is supposed to keep track of multiple registers and allow transferring funds between them. This means an end user can for example define registers: “wallet”, “savings”, “food expenses”, “car maintenance” and be able to track their spending by transferring amounts between them. In the future it would be possible to keep history of individual transfers, archive transfers and reset balance of registers – for now, however, this is out of scope and the business has pinpointed what they expect the app to be capable of for the sake of a successful presentation.

Business requirements

Business deems the following aspects as must-haves:

1. A demo environment with the following registers already existing:
   a. “Wallet” register with a balance of 1000
   b. “Savings” register with a balance of 5000
   c. “Insurance policy” register with a balance of 0
   d. “Food expenses” register with a balance of 0
2. A deployed application which needs to expose the following operations:
   a. Recharge an existing register with given amount – the register’s balance should then be updated accordingly with the requested amount added
   b. Transfer given amount between two existing registers – this means that given amount should be subtracted from source register’s balance and added to destination register’s balance
   c. Get current balance of all registers – this should simply inform about current balance of all existing registers
3. Data persistence: the application should persist balance of each register each time it is updated and even in case the whole system is turned off, all registers should have their balances set to previous values upon its restart.

Technical requirements

The following guidelines have to be taken into account:

1. The prototype is expected to be built on top of Spring Boot 2 framework
2. Persistence layer should adhere to JPA specification
3. It is expected that only endpoints mentioned above will be implemented
4. Justified usage of external libraries is permitted
5. A project README file describing how the app works and how to launch it

Example

The following scenario demonstrates what is the expected behaviour of the system. It assumes we are running a fresh demo (with all predefined registers existing and having corresponding balances):

1. A recharge is executed for the "Wallet" register with an amount of 2500.This should increase the register's balance to 3500.
2. A transfer of 1500 from "Wallet" to "Food expenses" registry is executed. This should bring "Wallet" balance to 2000 and "Food expenses" balance to 1500.
3. A transfer of 500 from "Savings" to "Insurance policy" registry is executed. This should bring "Savings" balance to 4500 and "Insurance policy" balance to 500.
4. A transfer of 1000 from "Wallet" to "Savings" registry is executed. This should bring "Wallet" balance to 1000 and "Savings" balance to 5500.
5. Balance info on all registries is executed: this should print the list of all registries accompanied by their balance, for example:

```
Wallet: 1000
Savings: 5500
Insurance policy: 500
Food expenses: 1500
```

Assessment scope

The following aspects of the solution will be assessed:

1. Architecture and API design
2. Choice of external libraries
3. Test coverage
4. Fulfilment of business and technical requirements
5. Complexity of the solution

Hints

The following hints might be helpful when doing the assignment:

1. Keep it as simple as possible
2. Instead of spending time implementing improvements or other features that have not been described as mandatory, it's better to just document them in the code or the README file
3. Focus on back-end only, front-end part is not required
4. Committing your work frequently to document the progress will allow us to follow your thinking and help understand what challenges you faced and how you chose to solve them

Publishing the solution

Please, publish your solution on your GitHub account and share just a link to point us to your repository.