Pile Coding challenge

Context

At Pile we work with financial data. Our customers leverage Pile products built on aggregating financial data from their multiple accounts. Our Product portfolio consists of two key products :

1. **Pile Account** : An interest bearing bank account that can be used to centralize all the banking features through a one unified interface.
2. **Multi-Banking** : Get a full view of your banking operations by connecting all your bank accounts in a single view.

Challenge

Our customers need to be able to transfer funds from their Pile bank account to other bank accounts. These transfers are done via SEPA bank transfer. For this coding challenge we ask you to build a basic SEPA transfer form. You are required to build a basic rest api. We will provide an account list in JSON format for the assignment. You are required to store the data in your db of choice. Please also include the data seeding step in your submission.

The functionality of the BE should be the following:

- List and filter the accounts based on the available balances (> x or <x). Implement pagination. Calculate the total balance for the filtered accounts.
- Get a single account based on the accounts IBAN
- Transfer funds between the accounts using the transaction form.
- Retrieve the transactions
- Log each transfer either through the application or your db extension of choice. Make sure that these logs persist.

Fields to include in the transaction form:

- Source account
- Amount input field formatted as EUR
- Recipient name
- Target IBAN
- Target BIC
- Reference

Technical requirements

Backend: Feel free to pick your stack here. We prefer Nodejs, Go or Rust. You should also containerize your application (either Podman of Docker).

FE: not required. However you are required to demonstrate how a FE would interact with BE. This can be a postman/insomnia collection, a list of curl scripts or a simple FE application.

Please send us a git repo link with clear instructions on how to test your work.

Evaluation

We are going to evaluate your submission using the following criteria:

- Ease of setting the application up and reproducibility
- Core functionality
- Handling of erroneous input
- DB utilization
- Code quality

Bonus points:

- Tests
- CI
- Caching
- Performance testing