Case study - Senior Backend Remote - Banking API

Created on	@Oct 09, 2019 5:19 PM
Duration	4 hours
Interviewers	
Teams	Engineering
Type	Coding (remote)

Could you be our next developer ?

We really appreciated our earlier discussions and want to continue the process further.

Our developers need to be experts at **designing the right solution** for any problem we have. As you would be a backend developper, your work will mostly be **giving our user the power over their financial data**, and this will be done through APIs.

This test has been made to give you an overview of some of the subjects you would work on, and we expect you to deliver a state of the art solution to the problem, including a documentation (yes, APIs come with documentation at Spendesk, think about the frontend team).

The problem

On Spendesk, customers have **wallets**, each of them representing a bank account in a specific currency**.**

Like with your own bank, each **card** you create will be connected to one of your wallets. Because the cards are prepaid, they need to be loaded :with money: before being used to pay on a merchant's website.

You may have guessed, **we need an API to manage the cards and the transfers between wallets.**

As we're actually moving customers money, we need to track any movement in the database carefully, especially who executed the transfer 👀

Specs

In the specs, a user represents a frontend client, the API should output the entities as JSON objects.

- A user can create a wallet in USD ($), GBP (£) or EUR (€)
- A user can create a card connected to one of his wallets
- A user can list all his cards
- A user can list all his wallets
- A user can load or unload money on his card from the wallet
- A user can block or unblock a card, blocking it will unload all the money into the right wallet
- Spendesk has a master wallet per currency where we store fees from transfers (see below)
- A user can transfer money between 2 wallets in different currencies but
- We take a 2.9% fee on the destination currency on this transfer
- This fee will go into our master wallet for the given currency
    - Of course you need to convert the amount (you can user **fixer.io** or any free API)
    - You don't need to manage users and authentication, just pass both a User-Id and a Company-Id headers with each request and use it to track

the money transfers / wallets or cards ownerships.

- As you can't use real money, you can load the wallet directly when you create it (setting the balance property), it does not work in real life but whatever!

Data structures

These are the mandatory fields for each entity, you can add as many as you want (even other entities if you need).

Wallet 💰

- Unique identifier
- Current balance
- Currency
- Company identifier
- Is master wallet (a boolean to know if it's ours for the fees)

Card 💳

- Unique identifier
- Wallet identifier
- Currency
- Current balance
- Random 16 digits number
- Expiration date (creation + 1 month)
- Random CCV (3 digits code on the back)
- User identifier (a card is associated to a user)
- Status (blocked or not)

Transfer

- Unique identifier

- Timestamp
- Amount transfered
- Origin currency
- Target currency
- Optional conversion fee
- Origin entity identifier
- Origin entity type (card or wallet)
- Target entity identifier
- Target entity type (card or wallet)

Expected output

You can write your code in any language and with any framework, we are mostly working with Javascript at Spendesk. Don't scratch your head too hard, a classic MVC architecture works well, just make sure to have everything ready so we can install your application and start testing it.

- Your source code, ideally on a Github repository but a ZIP works as well
- A getting-started guide, how to install your API, the technologies we may need (database, runtime...)
- A documentation of the endpoints

Bonus

- You handle the errors correctly on the API if we send random data
- You wrote some tests

Advice

- If you struggle on a specific point don't spend 2 hours on it, leave it aside and get back to it at the end
- If you have a question please send your contact an email

- This test is a real use case at Spendesk, put the same attention to this test that you would in your real work