PAYBACK Software Engineer (JAVA/SCALA) (m/f/d) – Machine Learning Engineering

Introduction

A central part of PAYBACK's business are coupons, which are assigned to members. Technically speaking this happens by storing the following information in a database:

- memberId (unique identifier of the PAYBACK member)
- couponId (unique identifier of the coupon)
- validFrom
- validUntil

When a member opens the PAYBACK App (or visits payback.de), he / she sees all coupons which have been assigned to him / her. Technically this happens by the App (or web frontend) making a " GetMemberCoupons" REST Call to the PAYBACK backend.

Coding challenge

Your task is to implement a web service, which exposes this "GetMemberCoupons" REST endpoint. To return a given member's coupons, the web service must query the database table (see description above) for the member's assigned coupons. You are free to choose whatever technology you like for the web service and the database (relational or NoSQL).

Requirements:

- You don't need to spend more than a few hours on your implementation. A basic running version (with one or two unit tests) is absolutely enough
- Only return coupons for the memberId, which is specified as a parameter in the REST call
- Only return coupons which are currently valid (i.e. the current date is between the validFrom and validUntil date)
- Sort the list of returned coupons by the validUntil date in descending order
- Keep performance and scalability in mind
- When you are finished, please push your code to a git repository and give us access

Notes

To keep it simple:

- Inserting data into the "assigned coupons" table can be done via script (e.g. shell, sql, ...). No need to implement a REST Endpoint for doing so.
- Implementing a frontend is out of scope. Calling the " GetMemberCoupons" endpoint via curl or the browser is sufficient.

Bonus (Optional)

If you want to challenge yourself even further, you can also implement this bonus feature:

- Each coupon is associated to a specified partner Branch location (e.g. Supermarket A in City B) represented by the latitude and longitude coordinates.
    - Add 2 columns to the coupon table to store the latitude and longitude information for each coupon
- Extend the parameters passed to the GetMemberCoupons Service to also include the current location of the member (in latitude and longitude)
- The GetMemberCoupons Service should now return a coupon list that is sorted by distance (nearest first).