CODING CHALLENGE – WEB SERVICES AND INTEGRATION

#CPM

ARCHITECTURE PRINCIPLES

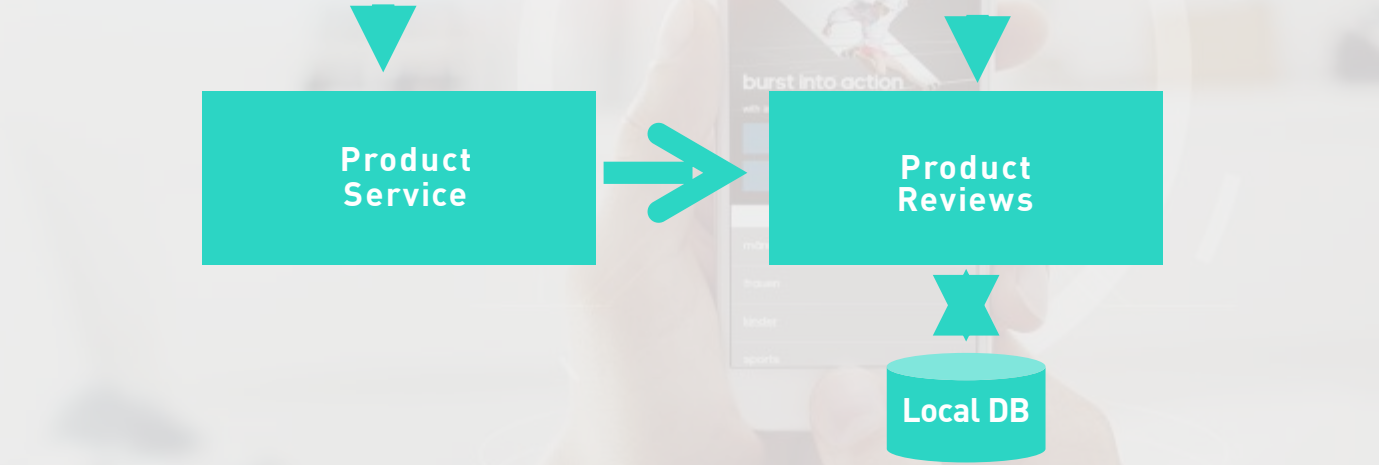
At adidas we care about **serving our customers** by making use of our data and services. Those services need to cope with **high volume** with **low latency**. In general, the topic of **high availability** is very important to be ready for the future and deal with the current state with confidence.

In a nutshell, the **basic architecture principles** are:

- Architect the solution in a way that makes them **reusable.**
- Always consider data & IT **security.**
- Solution should treat core operational database as a **costly** resource that needs to be used **economically.**
- Design for failure – your solution needs to be resilient by **handling exceptions.**

#CPM

ARCHITECTURE LANDSCAPE AND REQUIREMENTS



#CPM

MISSION STATEMENT

Primary tools: Java Spring Boot 2.X, REST API, Database
With the information given and additional assumptions of yours, you should develop **2 micro-services**:

1. **Review Service:**
   - implement CRUD operations for the resource:
     - **/review/{product_id}** , (e.g. AB1234), and the response is a JSON with the following data: **Product ID, Average Review Score, Number of Reviews.**
     - In order to protect the service and prohibit data tampering, authentication is needed to protect the write operations.
     - Choose any datastore for persisting the data – that can be easily deployed or installed with the application. The datastore should contain seeded data for a few products.
2. **Product Service:**
   - The service will expose the resource **/product/{product_id}**, only supporting GET. The response should be an aggregation* of our live product API, (*https://www.adidas.co.uk/api/products/{product_id}* e.g. **product_id** = C77154*)*, and the reviews retrieved from the Review Service for the same **product_id**
   - Sample product IDs: M20324, AC7836, C77154, BB5476, B42000, etc.

* aggregation: The JSON in the response should contain the fields retrieved from our live product API + the fields retrieved by the Review Service

#CPM

WHAT WE EXPECT FROM YOU

- Develop this application with a microservice approach, all services should run independently.
- Write API tests for your endpoints.
- Every person having Java and some standard tools should be able to check out the code, build and run the app locally (both Gradle and Maven accepted).
- Please, use English as a documentation language.
- When you are done, check your solution into any public GIT repo (e.g. GitHub or Bitbucket) and send us the link.

BONUS 1: Dockerize the two component services and create config files for deploying them.
BONUS 2: Create a CI/CD pipeline proposal for the app.
BONUS 3: Create a test integration suite for both services.

Now, finally, we don't limit you on anything. If you feel uncomfortable working with something, skip it. Alternatively, if you have an idea for a feature feel free to implement it within this challenge.

#CPM