NETWERVEN®

The Vacancy Repository problem

Description

You work in a project that is importing vacancies from several sources such as external API or an in house data warehouse. Because the import process is expensive you have already set up a local mysql database to store the data after the import. Additionally you might have a cache database (ie redis) for quicker access to frequently used data, or an elasticsearch database to quickly filter that data. The vacancy model in its simplest form has a description, a title, and some meta data associated with it. You can find a simple version of the vacancy model inside the current folder.

Requirements

(The number of hours after the requirements are an indication of the time you need to finish the task. If you need a lot of extra hours that's okay, the test is designed for senior developers).

Requirement 1 (4 hours).

Maintaining the code that satisfies the above conditions is obviously going to be a difficult task, unless there is a proper design to reduce complexity and enhance flexibility.

You are requested to design an architecture where all vacancy data will be requested from a centralized point; a class called VacancyRepository.

The VacancyRepository is responsible to talk to the different data sources, and for that reason it needs to be able to talk to a new data source with minimal additions to the code.

It also needs to be able to select which data sources is going to communicate with (in other words, the data sources should be able to be added/removed dynamically).

Requirement 2 (1 hour, can be combined with above).

Your implementation needs to be **accompanied by automated tests**.

Requirement 3 (extra, not necessary. 8 hours).

After the implementation of the above requirements you can now access your Vacancy model without caring where it came from. There is one little catch though. If you decide to update/delete the model, this action needs to be reflected to all the data sources that contain this vacancy. For that reason you are requested to design the appropriate classes/code to reflect the actions that alter a model to all associated data sources.