THE VIRTUAL FORGE®

Hi Anthony,

The Virtual Forge is excited to announce the next step of the recruitment process for .NET Developer.

We appreciate all the time spent with us.

To continue the process of becoming a VF employee, you'll have to complete the below Task.

Follow the following steps:

- Read carefully the exercise below, if you have any doubt don't hesitate in asking;
- Give us an estimated date for completing the task.

Any question you may have, feel free to send it to us. We will provide you with an answer as soon as possible.

Send your outcome to us by email.

Looking forward to hearing from you again.
Lots of luck!

Recruitment Team
The Virtual Forge

THEVIRTUALFORGE®

Musicalog - Music Store Catalog Web App

Functional Requirements

Musicalog is a web application that allows users to keep a catalogue of music albums. You are being asked to implement the 1st stage of the project, a web api that allows to list, create, edit or delete albums from the catalog.

Musicalog investors dream is to grow the system, creating UIs for mobile and web apps. They envision turning Musicalog into a cloud based software as a service. For this reason they take performance and scalability seriously. Also one of the investors is a phD in computer science, so complying with the best architectural practices is a must.

Album List Screen

A list of all the properties an album must have:

- Title;
- Artist Name;
- Type (vinyl/CD);
- Stock;

The web api must provide endpoint that allow the following actions:

- Retrieving a list of albuns. This list must be filterable by Title and Artist Name;
- Create a new album;
- Edit an existing album;
- Delete an existing album.

THEVIRTUALFORGE®

Non-Functional Requirements

The web api should be implemented using .NET CORE (3.1 or above).

There should be a clear separation between the application layers. The appropriate architecture and design patterns should be used.

The solution must have a project where the api endpoints are tested (unit testing or a simple console app).

The **data model should be relational and implemented on Microsoft SQL Server**. The data access technology (entity framework, dapper or other) should be chosen taking performance into consideration. The data model should be defined first, so **using a CODE FIRST approach to DB modelling IS NOT an option for this tech challenge**.

The web api code should be simple and easy to read. Adding new functionality or endpoints should be simple, all endpoints should follow a similar pattern.

Use an appropriate architecture, make sure you implement appropriate abstraction levels that help you replace components from your architecture in the future.

Remember that you are implementing an MVP and that if things go well the app will grow, so **you're about to implement what could be the foundations of a big building**.

Have fun.