Fullstack Code Challenge

Scenario

You have recently joined a company as a contractor, where they would like you to begin working on a new application which will allow them to keep track of and manage their employees.

The company has already created an empty Visual Studio solution and added some empty C# projects to represent a common n-tier architecture; your task is to fill in the code to meet the requirements provided below.

Guidelines

- The aim of this test isn't to complete every requirement as quickly as possible. We want to see how you solve particular problems so that we can discuss your solutions after the test.
- Please make use of any design patterns and principles you think are relevant and may show off your skills, but be prepared to discuss why you've used them!
- Front-end work is not required for this test, please only implement from the Controller / API endpoints down to the domain.
- You have **up to three hours** to do as much as you can. Please let us know when you would like to stop.

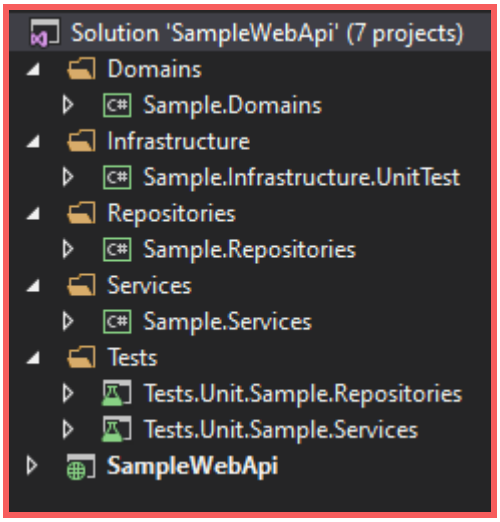
Click here to download source code

↓ Please scroll down ↓

Project Structure Overview

The project contains the following projects:

- **Sample.Domains;** contains all classes to represent business entities
- **Sample.Repositories;** contains all classes to abstract data access for the business entities
- **Sample.Services;** contains all classes to handle any business logic for entities
- **Unit Tests;** these contain all classes to define the unit tests for the various layers:
    - **Tests.Unit.Sample.Repositories**
    - **Tests.Unit.Sample.Services**
    - **Sample.Infrastructure.UnitTest** contains utility classes to help write tests
- **SampleWebApi** contains classes to provide a RESTful API for the front-end to call



Requirements

1). Domain Project

1. Update the **Employee** class and add the following properties:
    - **Id**; it's a required key property, and must be unique
    - **First Name**; it's a required property
    - **Last Name**; it's a required property
    - **Age**; it's a required property
    - **Gender**; it's a required property
    - **Full Name**; this should be **First Name** and **Last Name** combined, and should not be stored in the database

2). Repository Project

2. Update the **EmployeeRepository** class and implement the following methods (using Entity Framework):
    - **Insert**; this should store new employee into the database
    - **Update;** this should update an existing employee in the database
    - **Delete**; this should delete an existing employee from the database
3. Create a new DBContext class, which will allow access to the Employees database table via the EmployeeRepository.

3). Service Project

4. Update the **EmployeeService** class and implement the following methods:
    - **Save**; this should upsert an Employee, depending on whether it already exists in the database or not
    - **GetAll;** this should return a collection of all employees in the system, and allow filtering via first name, last name and gender

4). Tests.Unit.Sample.Services Project

5. Update the **EmployeeServiceTest** class and add some unit tests for the following methods:
    - **EmployeeService.Save**
    - **EmployeeService.GetAll**
    - Please feel free to implement mocking using any framework that you know; the **Sample.Infrastructure.UnitTest.MockDbSetHelper** class may help get you started.

5). Tests.Unit.Sample.Repositories Project

6. Update the **EmployeeRepositoryTest** class and add some unit tests for the following methods:
    - **EmployeeRepository.Delete**

6). Web API Project

7. Update the **EmployeeController** class and implement the following API endpoints:
    - **Save**; this should call the service to add or update an Employee, and then return a useful response to the front-end
    - **GetAll**; this should call the service to retrieve a list of Employees in the system, allowing any relevant filter arguments to be passed in from the front-end
    - Please ensure that all:
        i. Inputs are validated and sanitized
        ii. Requests are authenticated