# Компетенции сотрудников (Hall Of Fame)

Представьте, что вы работаете в IT-компании, где множество сотрудников владеют разными навыками на разных уровнях. Сотрудники периодически повышают уровень существующих навыков, овладевают новыми навыками, и руководитель хочет видеть актуальные компетенции команды. Вам дают задание разработать backend-часть одностраничного приложения для просмотра и редактирования навыков персонала.

## Обязательные требования

1. На стороне backend нужно использовать ASP.NET Core актуальной версии.
2. База данных: MS SQL ORM: EF Core. Структура БД должна приводиться к кодовой базе при помощи миграций.
3. Программный код должен быть оформлен по одному из общепринятых соглашений по оформлению кода. Например, <https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/inside-a-program/coding-conventions>.

## Основные сущности

Система должна оперировать следующими сущностями:

**Person (Сотрудник):  
{**

**id: long,**

**name: string,**

**displayName: string,**

**skills: [Skill, Skill, Skill, …]**

**}**

**Skill (Навык)**

**{**

**name: string,**

**level: byte // 1-10**

**}**

## API взаимодействия

**GET api/v1/persons**

Возвращает массив объектов типа Person:

[Person, Person, …]

**GET api/v1/person/[id]**

Где id – уникальный идентификатор сотрудника.

Возвращает объект типа Person.

**POST api/v1/person**

Где id – уникальный идентификатор сотрудника.

В теле запроса передавать объект Person. Id должен быть null или undefined.

Создаёт нового сотрудника в системе с указанными навыками.

**PUT api/v1/person/[id]**

Где id – уникальный идентификатор сотрудника.

В теле запроса передавать объект Person. Id должен быть null или undefined.

Обновляет данные сотрудника согласно значениям, указанным в объекте Person в теле. Обновляет навыки сотрудника согласно указанному набору.

**DELETE** **api/v1/person/[id]**

Где id – уникальный идентификатор сотрудника.

Удаляет с указанным id сотрудника из системы.

Сервер должен уметь отслеживать изменения в навыках сотрудника при сохранении.

### Статусы ответов:

200 – успешное выполнение запроса.

400 – неверный запрос.

404 – сущность не найдена в системе.

500 – серверная ошибка (например, при обработке данных).

# Если задание показалось слишком простым

…или хочется получить дополнительные практические навыки. Есть ещё ряд бонусных требований:

1. Сервисы, работающие с бизнес-логикой, должны быть покрыты модульными и интеграционными тестами.
2. Backend должен предоставлять интерфейс Swagger.
3. Система должна корректно обрабатывать ошибки пользовательского ввода. Сбои при работе приложения должны логироваться в физический файл.