Тестовое задание

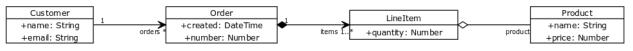
Бэкенд-разработчик (ASP.NET)

Описание задачи

Реализовать бэкенд-приложение простой ecommerce-системы, которое позволяет:

- 1. Проводить CRUD-операции для Продуктов.
- 2. Создавать Заказы
 - а. В процессе создавать Клиентов (данные клиента приходят в запрос вместе с описанием заказа).
 - b. Если заказчик с указанным емейлом существует, создавать Заказ для этого Клиента
- 3. Запрашивать следующие данные:
 - а. Список Заказов для указанного Клиента с указанием общей стоимости каждого.
 - b. Список Продуктов, отсортированные по популярности (количество уникальных заказов, в котором он участвует), для каждого продукта должно быть указано общее количество проданных единиц.
 - с. Список Клиентов, заказавших товара на сумму, превышающую указанную.

Следующая диаграмма классов иллюстрирует структуру данных:



CREATED WITH YUML

Ограничения предметной области

- 1. Номер заказа уникален.
- 2. В одном заказе может быть несколько Позиций с разными продуктами. Нельзя создавать несколько Позиций с одинаковыми продуктами в рамках одного заказа.
- 3. Количество продуктов в одной Позиции должно быть положительным числом.

- 4. Цена Продукта является неотрицательным числом. Имя Продукта уникально.
- 5. Емейл Клиента-уникален в рамках системы.

Технические требования

Требование к приложению

- 1. Приложение необходимо реализовать на базе .NET Core 3.1 на языке C# в виде Web API.
- 2. Описанные выше операции должны выполняться HTTP-запросами (GET, POST, PUT, DELETE и т.п.). Данные должны возвращаться в формате JSON.
- 3. Подключить Swagger-клиент (например, используя Nuget-пакет Swashbuckle).

Требования к работе с данными

- 1. Хранение данных следует реализовать в базе данных MSSQL(или любой другой СУБД).
- 2. Доступ к данным через Entity Framework C использованием подхода Code First.
- 3. Создание Заказа необходимо производить единой транзакцией. Исключения при создании Заказов не должны оставлять неконсистентные данные.
- 4. Когда возможно, требуется применять асинхронные операции.

Мы будем обращать внимание не только на работоспособность приложения, но и на его архитектуру. Поэтому крайне приветствуется применение подходящих паттернов проектирования (например, Repository, Unit of work и т.п), разделение приложения на слои (предметная область, веб-приложение и т.п) и прочим практикам, которые вы бы применили, если бы это был реальный проект «настоящего» масштаба. Наличие юнит-тестов, запуск из Docker и т.д. приветствуются, хотя являются опциональными.

Результат в виде .zip-файла или ссылки на Github отправляйте на адреса <u>m.smalekha@aurigma.com</u> (Максим Смалёха) и <u>andrew@aurigma.com</u> (Андрей Симонцев). Любые вопросы по заданию можно направлять по тем же адресам, либо по телефону 26-37-73.