Тестовое задание для разработчика Backend C#

# Проблематика

Компании требуется Web-приложение мониторинга и управления электронными счетами клиентов, которое позволит просматривать все уже выставленные счета, а также создавать новые счета. Для реализации этого приложения требуется создать два решения – web-сервис, который предоставляет API для работы со счетами и браузерное web-приложение для предоставления UI управления счетами.

# Функциональные требования

Данное задание ограничивается реализацией web-сервиса, предоставляющего API.

## Структура данных

Счет:

* Дата и время создания счета;
* Дата и время изменения счета;
* Номер счета;
* Статус обработки счета (при создании устанавливается в значение «Новый»);
* Сумма счета;
* Способ оплаты счета.

Статусы обработки счета:

* 1 – Новый;
* 2 – Оплачен;
* 3 – Отменен.

Способы оплаты счета:

* 1 – Кредитная карта;
* 2 – Дебетовая карта;
* 3 – Электронный чек.

## Хранение данных

Для хранения данных используется файл CSV Invoices.csv. При сохранении нового счета этот счет должен быть добавлен в файл без повреждения уже присутствующих в файле счетов.

## API

API должен предоставлять средующие функции:

1. Получение всех счетов в системе с постраничным разбиением.

Входные данные: принимается фильтрация, порядок сортировка и параметры страницы.

Выходные данные: страница счетов.

1. Получение одного счета по идентификатору.
2. Добавление счета.
3. Изменение счета.

При добавлении и изменении счета должна выполнятся проверка данных: номер должен быть заполнен, сумма должна быть заполнена и быть числом. В случае неуспешной проверки API возвращает ошибку. В случае успешной проверки счет сохраняется.

# Нефункциональные требования

1. Решение должно влючать в себя API работы со счетами на .NET Core.
2. API должен быть оформлен как решение для Visual Studio.
3. Код API должен быть написан на языке C# и соответствовать лучшим практикам программирования (например, рекомендациям Microsoft для .NET).

# Дополнительные задачи

Также приветствуется, если:

1. Переключить использование хранилища с файла со счетами на Entity Framework code first.
2. Добавить SPA UI на одной из современных платформ: Angular, ReactJS, Vue.js.

# Заключение

Тестовое задание является упрощенным вариантом реальных задач компании, поэтому при оценке его решения будет учитываться:

* Насколько аккуратно написан код;
* Насколько подходящими являются выбранные наименования объектов;
* Насколько хорошо разделены зоны ответственности объектов;
* Насколько легко будет поддерживать в дальнейшем это решение.