Тестовые задания на позицию Middle Fullstack C# Developer в компании «ООО СМАРТ МИЛ СЕРВИС»

Выполните два практических задания ниже.

Если в заданиях что-то непонятно, опишите возникшие вопросы и сделанные предположения, например, в комментариях в коде.

По завершении работы разместите код на *GitHub* и отправьте ссылку Рекрутеру.

Задание №1

Часть 1

Имеется некий сервер, принимающий запросы по протоколу *HTTP* с *Basic*-аутентификацией.

Для всех запросов используется один общий endpoint.

Сервер всегда возвращает ответ с кодом 200, если запрос был выполнен (даже если тело запроса некорректно). В случае ошибки тело ответа будет содержать поля *Success = false* и *ErrorMessage* с описанием текста ошибки.

Требуется написать библиотеку (компилируемую в файл вида ***<название>.dll***), которая должна выполнять следующие задачи:

1. Отправлять на сервер *Запрос №1*, парсить ответ (при условии, что сервер вернул *Success = true*) и возвращать массив ***Блюд***. Примеры запроса и ответа приведены в Приложении №1.
2. Отправлять на сервер *Запрос №2*, в котором передавать ***Заказ***. Примеры запроса и ответа приведены в Приложении №2.

Сущности ***Блюдо*** и ***Заказ*** требуется предварительно описать в виде классов.

Каждую задачу должен решать отдельный метод.

Название метода, список параметров и тип возвращаемого значения Исполнитель определяет самостоятельно.

Допускается использование популярных библиотек для работы с *JSON*.

Часть 2

Вместо прямого обмена по протоколу *HTTP* использовать обмен с использованием фреймворка *gRPC* (без авторизации). Прото-файл с описанием передаваемых и получаемых данных представлен в Приложении №3.

Часть 3

Создать консольное приложение на базе платформы *.NET* 8.0. Подключить к нему *DLL*-библиотеку, созданную ранее (какую библиотеку подключить Исполнитель определяет на свое усмотрение). Приложение должно выполнять следующие задачи (последовательно):

1. Инициализировать новую БД и таблицу в ней для хранения данных, при условии, что БД и/или таблица не существуют (предпочтительно использовать СУБД PostgreSQL).
2. Вызвать первый метод из библиотеки для получения списка ***Блюд*** от сервера.
3. Записать результаты выполнения метода в таблицу в БД, а также вывести список ***Блюд*** в консоль в следующем формате:

   Название – Код (артикул) – Цена за единицу

   Если от сервера был получен негативный ответ, вместо этого вывести текст сообщения об ошибке. Выполнение программы в таком случае прекращается.
4. Создать экземпляр класса ***Заказ***.
5. Предоставить возможность ввода списка ***Блюд*** с клавиатуры одной строкой в следующем формате:

   Код1:Количество1;Код2:Количество2;Код3:Количество3;…
6. Проверить, что все введенные коды существуют, а указанное количество для всех позиций больше нуля. В случае обнаружения некорректного ввода уведомить об этом пользователя и предоставить возможность повторного ввода.
7. Добавить в ранее созданный ***Заказ*** введенные позиции и отправить их на сервер путем вызова из библиотеки второго метода.
8. Вывести в консоль ответ от сервера: в случае успеха написать слово ***УСПЕХ***, в случае неудачи – текст сообщения об ошибке.

Дополнительные условия:

1. Подключить конфигурационный файл ***appsettings.json***, в котором описать строку подключения к БД и другие данные на усмотрение Исполнителя.
2. Все содержимое консоли должно записываться в файл логов, формат названия следующий:
   test-sms-console-app-yyyyMMdd.log

Для чтения конфигурационного файла и логирования допускается использование популярных библиотек.

Задание №2

Создать десктопное *WPF*-приложение на базе платформы *.NET* 8.0. Приложение должно предоставлять возможность чтения и изменения некого набора переменных среды в графическом формате.

Внешний вид *UI* должен соответствовать изображенному в Приложении №4, но *Pixel-perfect* верстка при этом не требуется.

Вводимые Пользователем значения переменных могут быть только текстовые и на практике не иметь ограничений по длине.

При запуске производится чтение переменных и вывод их на экран, либо инициализация значениями по-умолчанию, если какая-либо из переменных не существует.

Все записанные переменные и их значения должны быть доступны другим приложениям и сохраняться даже после перезапуска ОС.

Дополнительные условия:

1. Подключить конфигурационный файл ***appsettings.json***, в котором описать набор имен используемых переменных среды в виде массива строк.
2. Все факты изменения переменных среды должны фиксироваться в файл логов, формат названия следующий:
   test-sms-wpf-app-yyyyMMdd.log
   Формат самих сообщений устанавливает Исполнитель на свое усмотрение.

Для чтения конфигурационного файла и логирования допускается использование популярных библиотек.

Пример запроса №1:


```
{
  "Command": "GetMenu",
  "CommandParameters": {
    "WithPrice": true
  }
}
```


Пример ответа на запрос №1:


```
{
  "Command": "GetMenu",
  "Success": true,
  "ErrorMessage": "",
  "Data": {
    "MenuItems": [
      {
        "Id": "5979224",
        "Article": "A1004292",
        "Name": "Каша гречневая",
        "Price": 50,
        "IsWeighted": false,
        "FullPath": "ПРОИЗВОДСТВО\\Гарниры",
        "Barcodes": [
          "578909756279742364 29"
        ]
      },
      {
        "Id": "9084246",
        "Article": "A1004293",
        "Name": "Конфеты Коровка",
        "Price": 300,
        "IsWeighted": true,
        "FullPath": "ДЕСЕРТЫ\\Развес",
        "Barcodes": []
      }
    ]
  }
}
```

Пример запроса №2:


```
{
  "Command": "SendOrder",
  "CommandParameters": {
    "OrderId": "62137983-1117-4D10-87C1-EF40A4348250",
    "MenuItems": [
      {
        "Id": "5979224",
        "Quantity": "1"
      },
      {
        "Id": "9084246",
        "Quantity": "0.408"
      }
    ]
  }
}
```


Пример ответа на запрос №2:


```
{
  "Command": "SendOrder",
  "Success": true,
  "ErrorMessage": ""
}
```

```
syntax = "proto3";

import "google/protobuf/wrappers.proto";

package sms.test;

option csharp_namespace = "Sms.Test";

message MenuItem {
  string id = 1;
  string article = 2;
  string name = 3;
  double price = 4;
  bool is_weighted = 5;
  string full_path = 6;
  repeated string barcodes = 7;
}

message OrderItem {
  string id = 1;
  double quantity = 2;
}

message Order {
  string id = 1;
  repeated OrderItem order_items = 2;
}

message GetMenuResponse {
  bool success = 1;
  string error_message = 2;
  repeated MenuItem menu_items = 3;
}

message SendOrderResponse {
  bool success = 1;
  string error_message = 2;
}

service SmsTestService {
    rpc GetMenu(google.protobuf.BoolValue) returns (GetMenuResponse);
    rpc SendOrder(Order) returns (SendOrderResponse);
}
```

Приложение №4

Рисунок 1 – Внешний вид UI для верстки