Необходимо разработать систему с использованием ASP .NET Core 3.1 и Entity Framework.

В системе должен быть контроллер, который принимает заказы и сохраняет их в базу данных. Контроллер должен принимать POST запросы, с параметром в котором указан из какой системы заказ и в теле запроса JSON данные о заказе.

Пример URL /api/order/talabat, /api/order/zomato, /api/order/uber. Где talabat, zomato, uber - это различные системы.

Структура заказа в базе данных имеет вид:

| id | system\_type | order\_number | source\_order | converted\_order | created\_at |
| --- | --- | --- | --- | --- | --- |
| уникальные идентификатор | тип заказа (zomato, uber, talabat) | номер заказа из поля orderNumber | исходный JSON заказа полуенный через контроллер | JSON заказа после конвертации | дата создания заказа на сервере |

Пример Json заказа имеет следующую структуру:

{

"orderNumber":"352225243",

"products":[

{

"id":"",

"name":"Papas House Pasta",

"comment":"no green peppers, no pepperoni and no Turkey ham. Instead add grilled chicken",

"quantity":"1",

"paidPrice":"24",

"unitPrice":"24",

"remoteCode":"ProdId\_33bf1315-3c0c-4df5-952b-f45a5cb1e501",

"description":"",

"vatPercentage":"",

"discountAmount":""

},

{

"id":"",

"name":"Cheesesticks",

"comment":"",

"quantity":"1",

"paidPrice":"21",

"unitPrice":"21",

"remoteCode":"ProdId\_ddb762ea-5dd4-4a40-bdda-63f699c38aa3\_77aa67ad-37bc-4ee7-8d2a-4320e3c04098",

"description":"",

"vatPercentage":"",

"discountAmount":""

},

{

"id":"",

"name":"Tiramisu",

"comment":"",

"quantity":"2",

"paidPrice":"30",

"unitPrice":"30",

"remoteCode":"ProdId\_70bbd461-bc1c-49fa-a23e-0718d8536284",

"description":"",

"vatPercentage":"",

"discountAmount":""

},

{

"id":"",

"name":"Pepperoni Rolls",

"comment":"",

"quantity":"1",

"paidPrice":"23",

"unitPrice":"23",

"remoteCode":"ProdId\_27d21751-b288-4682-b0d4-0dee7284d7ed\_5eaee492-c695-4657-8ad7-d1fdf0ac0c24",

"description":"",

"vatPercentage":"",

"discountAmount":""

}

],

"createdAt":"2020-11-20T18:21:19.069+03:00"

}

Необходимо разработать сервис (класс) который, получает из базы данных новые/необработанные заказы и обрабатывает их. Сервис должен запускаться с заданной периодичностью в 5 секунд.

В зависимости от типа заказа должен быть вызван специфический код для каждой системы: talabat, zomato, uber. Важно: Необходимо учесть что в будущем типов систем может быть больше 3, поэтому нужно решение которое будет работать не через if else или swich case, чтобы не нужно было менять код в этом сервисе, при добавлении нового обработчика для новой системы.

После обработки необходимо изменить статус заказа на обработанный, обновить измененный JSON заказа и сохранить изменения в базе данных (в поле converted\_order).

В случае возникновения исключения, необходимо отправить текст ошибки в сервис отвечающий за уведомление о сбоях (см. далее). При вызове сервиса необходимо предусмотреть что его работа может быть долгой, а сервис отвечающий за обработку заказов не должен прерываться и дожидаться окончания его работы.

Код обработчиков

talabat - принимает JSON заказа и меняет все положительные цены в заказе на отрицательные. Возвращает измененный заказ. (Продукты которые нужно обработать содержаться в коллекции products. Цены содержаться в поле paidPrice)

zomato - принимает JSON заказа и делит все цены в заказе на количество позиций (price / quantity). Возвращает измененный заказ (Продукты которые нужно обработать содержаться в коллекции products. Цены содержаться в поле paidPrice, количество в поле quantity)

uber - принимает JSON заказа и выбрасывает исключение

Сервис (класс) уведомления о сбоях

Принимает описание об ошибке и пишет ее в лог-файл.

Делает задержку в 10 секунд после каждой записи, имитируя долгую работу.

**Примечание.**

Под "Сервис" в тексте подразумевается просто класс Service, а не отдельное решение/служба и т.п.

Все должно быть реализовано в рамках одного проекта (либо нескольких проектов, но в едином Solution)

Соответственно не нужно использовать какие-то сторонние решения (например MQ), чтобы устанавливать взаимодействия между решениями.