**ТЕСТОВОЕ ЗАДАНИЕ – СЕРВИС СООБЩЕНИЙ**

Необходимо реализовать онлайн-сервис отправки сообщений, удовлетворяющий следующим требованиям:

1. Технологии: ASP.NET WebAPI, ASP.NET MVC, Entity Framework (MS SQL), исходники разместить в публичном репозитории на github.
2. Сообщения могут быть трёх видов: email, sms, push. Отправку всех трёх видов эмулируем отправкой email на указанный в настройках (web.config) адрес, с указанием в тексте письма какой это тип сообщения и текста сообщения.
3. Проект должен состоять (как минимум) из двух частей (проектов): WebApi сервиса и web-интерфейса на ASP.NET MVC, который вызывает методы сервиса (HttpClient) (вызовы методов api делать со стороны сервера, а не из браузера, но при этом ограничений на использование ajax для UI-целей нет).
4. WebApi должен иметь следующие методы:
   1. send-message (POST) – помещает сообщение в очередь отправки сообщений. В качестве параметров принимает:
      1. Текст сообщения
      2. Какими каналами отправить сообщение (email, sms, push). Можно указать несколько вариантов сразу.
   2. queue (GET) - получить очередь сообщений. Может принимать параметры, если в этом есть необходимость (на усмотрение разработчика).
   3. queue-step (GET) – проверяет очередь сообщений и отправляет одно из сообщений (интересно какое? см. п.7.). Не принимает параметров.
5. Web-интерфейс должен иметь все необходимые страницы для выполнения следующих задач:
   1. Отправка сообщения с указанием способов доставки (email, sms, push).
   2. Просмотра очереди сообщений (предусмотреть пагинацию).
   3. Выполнения шага движения очереди (вызов queue-step).

Подумать, как будет удобнее этим всем пользоваться, какие роли «пользователей» в системе и, исходя из этого, выбрать количество и состав страниц.

1. Предусмотреть логирование успешных и неуспешных операций на стороне сервиса (использовать log4net).
2. Предусмотреть случаи сбоев при оправке сообщений. Выбрать стратегию поведения в случаях, если отправка какого-либо сообщения не удалась, и реализовать выбранную стратегию.

## Критерии качества выполнения задания

1. Архитектурная целостность. Решение должно быть логичным, полноценным, использовать адекватные решения и иметь чёткую структуру слоев и классов.
2. Понятность. Рассматриваем решение так, чтобы другой разработчик мог в будущем быстро разобраться в коде и вести доработку.
3. Отсутствие избыточности. Не должно быть лишних внешних зависимостей, либо избыточных интерфейсов, применение которых не оправдано задачей.
4. Эффективность. Выжимать каждую миллисекунду не требуется, однако в результате не должно быть явных узких мест, на которых будут значительные потери эффективности исполнения кода.
5. Целесообразность. Не нужно изобретать велосипед заново в том случае, когда этого явно не требуется. Не нужно чрезмерно усложнять либо чрезмерно упрощать решения.
6. Красота. Мы не требуем специального дизайнерского оформления пользовательского интерфейса, однако он не должен выглядеть отвратительно.