**Демонстрация возможностей и ключевых элементов разработанного**

**в рамках тестового задания сервиса «Поздравлятор».**

Автор: Рупасов Кирилл

В рамках работы был разработан Веб сервис, состоящий из трех основных компонентов:

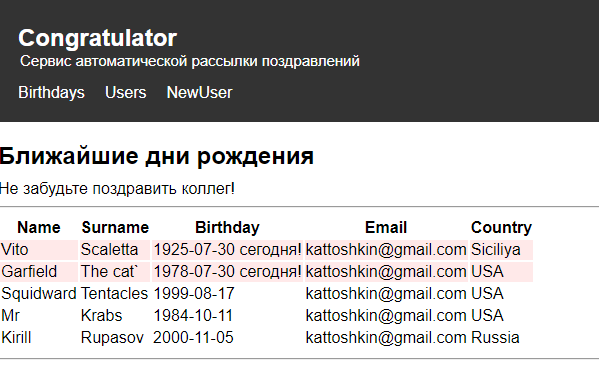
* Frontend Веб-интерфейс (**Angular SPA**)
* Backend API (**C#, ASP.NET Core** **Веб-API, Entity Framework, MS SQL Server**)
* Сервис для рассылки почты (**C#, SMTP**)

Также в качестве связующего звена между последними был выбран брокер сообщений **RabbitMQ**.

Ссылка на GitHub:

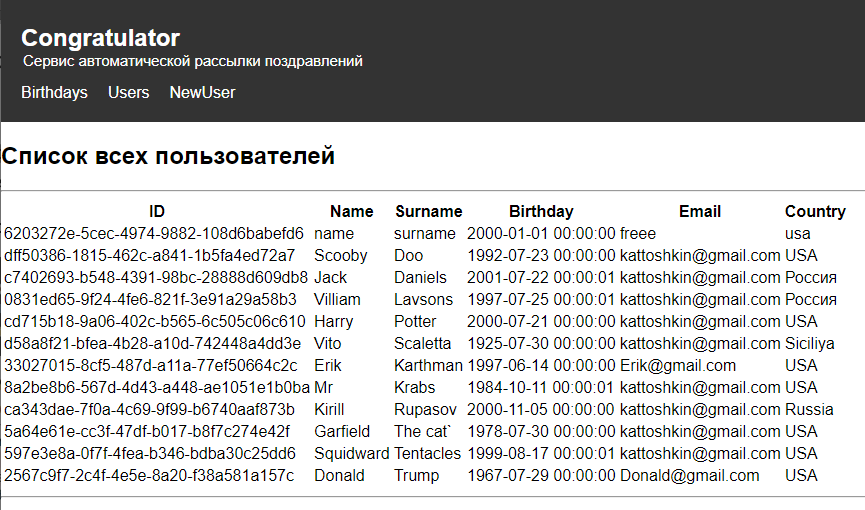
1. **(CongratulatorWebAppClient) Веб-интерфейс** представляет собой набор вью, позволяющих добавлять и отображать пользователей.

**Главная страница (Birthdays)** (ближайшие дни рождения)

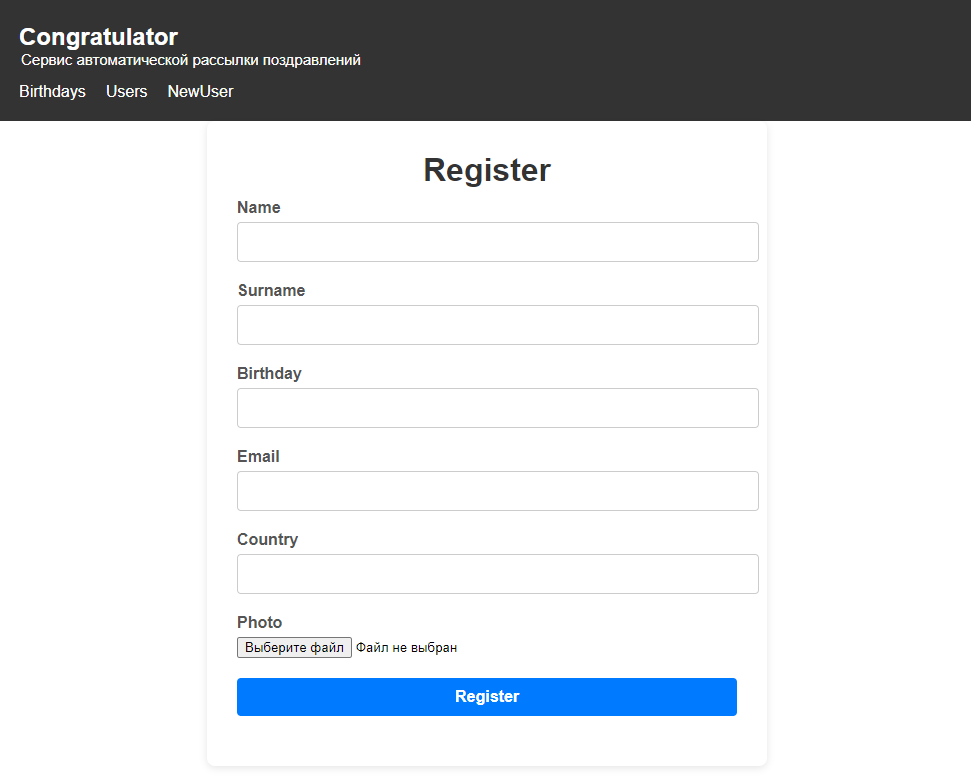


* Отображает 5 ближайших именинников, подсвечивая цветом сегодняшних.

**Страница отображения** полной информации о всех пользователях **(Users):**



**Страница регистрации** нового пользователя **(NewUser)**:



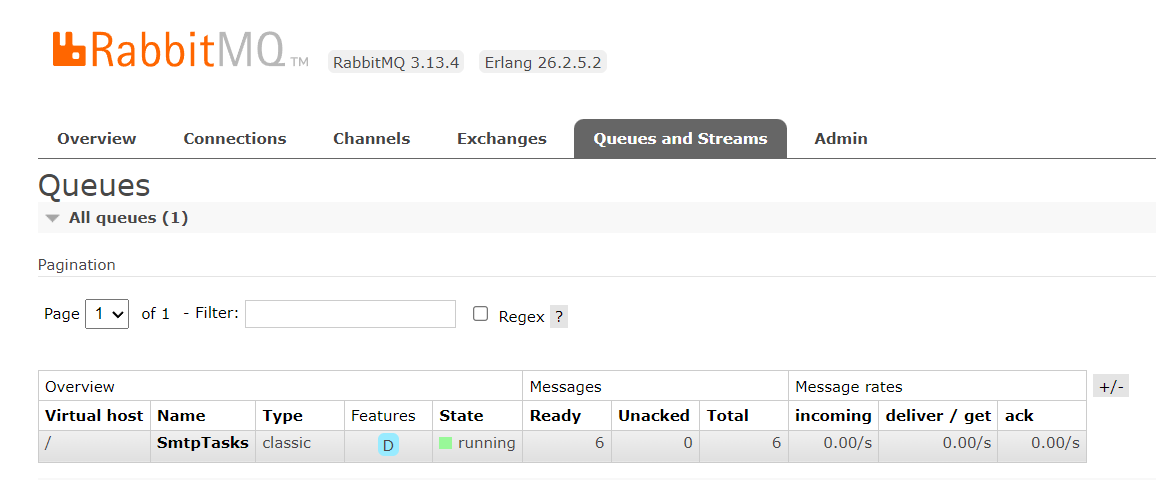
1. **(CongratulatorWebAppAPI) Backend API** представляет собой приложение ASP.NET Core Веб-API и:
   * Хранит основные сведения о модели сервиса (определение сущностей, Dto-объектов, …)
   * Реализует основную логику сервиса (Controllers, Scheduler проверки именинников, отправка заданий SMTP сервису на отправку писем, …)
   * Реализует контекст данных и подключение к базе
   * Открывает API для Frontend-приложения

Основные сущности:

* User – описывает сущность пользователя (Имя, Фамилия, Почта, Страна и Дата рождения)
* UserCongratulation – описывает сущность поздравления (Сообщение, Официальность)

1. **(SmtpServiceConsoleApp) Сервис SMTP** рассылки осуществляет рассылку поздравлений на почту. Работает в качестве фоновой службы, которая постоянно через Scheduler проверяет очередь RabbitMQ, откуда получает задания на отправку сообщений именинникам.

**RabbitMQ** разворачивается в Docker, на скрине представлен пример накопления сообщений в очереди для последующей рассылки по smtp.



Пример результата работы рассылки по smtp:

