Prueba técnica Digital Solutions

Índice

[Prueba técnica 3](#_heading=h.30j0zll)

[Especificaciones 3](#_heading=h.1fob9te)

[Usuarios 3](#_heading=h.3znysh7)

[Publicación de mensajes 3](#_heading=h.2et92p0)

[Follow 3](#_heading=h.tyjcwt)

[Dashboard 4](#_heading=h.3dy6vkm)

[Carga de archivo 4](#_heading=h.1t3h5sf)

[Requisitos Generales 4](#_heading=h.4d34og8)

# Prueba técnica

Implementar una aplicación de red social en modo consola (similar a Twitter). Se debe de poder realizar interacciones por medio de comandos de texto sencillos (frontend) hacia una web api (backend), estos comandos se describirán más adelante. Se deben de generar pruebas unitarias y cumplir con un code coverage del 80%. Se tendrá un archivo de texto el cual contendrá un conjunto de comandos de la aplicación, el cual servirá para comprobar el correcto funcionamiento de la misma. Todas las pruebas unitarias deberán de pasar satisfactoriamente.

## Especificaciones

Se debe de realizar en un entorno .NET (web api en backend y proyecto en consola para el frontend), las versiones quedan a su discreción.

## Usuarios

Se debe de poder crear usuarios.

* Por defecto existirán 3 usuarios ya creados, @Pedro, @Jorge, @Fernanda.
* Para crear nuevos usuarios se utilizará el siguiente comando
  + post #Usuario Luis
* Donde:
  + post: es la acción
  + #Usuario: indica que será una creación de usuario
  + Luis: es el nombre del usuario, no se aceptan espacios en blanco en el nombre.
* Si el usuario ya existe debe de indicar que ya existe y no se creara uno nuevo.

## Publicación de mensajes

Cada usuario debe de poder publicar mensajes en su muro, de manera que estos puedan ser visualizados más adelante.

* La forma de crear mensajes será la siguiente:
  + post @Luis Hola a todos
* Donde:
  + post: es el comando a utilizar
  + @Luis: es el usuario que lo publicara
  + Hola a todos: es el mensaje que se publicara en el muro de Luis.
* El sistema deberá de mostrar como resultado lo siguiente
  + Luis posted -> “Hola a todos” @10:30
* Otro ejemplo:
  + post @Jorge me siento feliz
* Resultado:
  + Jorge posted -> “me siento feliz” @17:20

## Follow

Cada usuario tiene la posibilidad de seguir a los usuarios existentes que desee.

* La forma en que se realizar es la siguiente:
  + Follow @Luis @Jorge
* Donde:
  + follow: indica la acción de seguir
  + @Luis: es el usuario que va a realizar la acción de seguir
  + @Jorge: es el usuario al cual Luis va a seguir.
* El resultado de este comando deberá de ser el siguiente:
  + “Luis empezó a seguir a Jorge”
* Si no existe alguno de los dos usuarios deberá de indicarlo:
  + “El usuario Luis no existe”
  + “El usuario Jorge no existe”
* Si Luis ya sigue a Jorge y vuelve a ejecutar la acción deberá mostrar lo siguiente:
  + “@Luis ya esta siguiendo a @Jorge”

## Dashboard

El dashboard es el muro donde cada usuario podrá visualizar el propio, por lo tanto, podrá ver publicaciones hechas por los usuarios a los que sigue, ordenados por hora.

* El comando es el siguiente:
  + dashboard @Luis
* Donde:
  + dashboard: es la acción a realizar
  + @Luis: es el usuario que quiere ver su muro
* El resultado es el siguiente:
  + “Hoy puede ser un gran dia” @Ivan @08:10
  + “Hola mundo” @Alfonso @10:30
  + “Para casa ya, media jornada, 12h” @Ivan @20:10
  + “Adiós mundo cruel” @Alfonso @20:30

## Carga de archivo

Se debe permitir cargar un archivo con comandos ya definidos, cada comando se realizará línea por línea.

* El comando a utilizar es el siguiente:
  + load “C:\archivo.txt”
* Donde:
  + load: indica que se cargara el archivo
  + “C:\archivo.txt”: indica el path a utilizar para cargar el archivo.
* Resultado:
  + “Se ha cargado con éxito el archivo archivo.txt”
* Si hubiera un problema con el archivo deberá mostrarlo:
  + “No existe el archivo archivo.txt”
  + “El archivo.txt está bloqueado”

## Requisitos Generales

* Nosotros tendremos un archivo de pruebas para cargar en el programa, sin embargo, se necesita que el realizador de la prueba genere uno propio el cual servirá tanto en la ejecución del proyecto como en los test.
* Se valorará la arquitectura y el uso de patrones en la aplicación.
* No es necesario preocuparse de comandos inválidos o excepciones, asume que el usuario siempre va a escribir bien los comandos y siempre con los parámetros correctos.
* No se requiere persistencia en bbdd, ni en disco, si se realiza será un plus.
* La prueba puede realizarse tanto en .Net Core como .Net Framework
* El lenguaje utilizado puede ser cualquiera soportado por .Net, pero se recomienda C# o Visual Basic.
* La prueba debe subirse a un repositorio GIT público.
* Se debe de enviar el enlace para poder visualizarlo.
* El proyecto se debe de poder ejecutar en Visual Studio Community 2022.