Taller de Programación GR 6

Semestre 01 – 2019

Integrantes:

Sebastián López Hererra

2019053591

Daniel Sequeira Retana

2019039641

Tarea Programada #2

Entrega: martes 14 de mayo de 2019

Estatus: Excelente

# Enunciado del proyecto:

Un cliente nos contrata con la esperanza de que le desarrollemos un programa de repositorio de frases célebres de ***Star Wars***, debido a que es un fanático de las películas.

El cliente nos solicita que las frases deben de provenir de un API que una amiga diseño,

este API se encuentra en la URL http://swquotesapi.digitaljedi.dk/index.html. Considere que el TEC, bloquea esta ruta de trabajo.

El cliente posee cierto conocimiento en el tema de la programación informática, por lo que

nos solicita las siguientes estructuras del sistema:

Estructura: Matriz

Él sabe que el API de su amiga retorna la información como texto plano, en el cual se

incluye un “Id” de la frase, además de la frase deseada y del personaje que la dijo.

El cliente quiere una matriz de la siguiente forma:

Nombre del personaje **Lista** de frases ***Lista*** de los id’s Código de aplicación

Esta matriz debe de expandirse y o modificarse cada vez que la llamada a la API se realice, insertando los datos obtenidos. De forma inteligente, el sistema debe saber si la frase retornada del API ya está en el sistema y no permitir duplicados.

Estructura: Diccionario

Además, quiere que el programa posea un diccionario, donde la llave es el nombre del

personaje y el contenido es una lista compuesta por:

***Código de la aplicación*** que el cliente especifico como: El símbolo “#” seguido por la primera letra del nombre del personaje, seguido por un número serial de 000 a

999, que debe crecer automáticamente, seguido por un “-”, terminando en la última

letra del nombre del personaje. Ejemplo, si el quinto personaje encontrado en la

aplicación es “Yoda” el código de la aplicación sería: #Y005-A

***Número de resultados que el API ha dado***, de ese personaje para propósitos

estadísticos. Aquí debe aumentarse el contador a pesar de aparecer frases

repetidas.

Compartir datos

Por último el cliente quiere poder compartir y guardar las frases que ya obtuvo, por lo que

piensa que la mejor forma de realizarlo es mediante archivos extensión “***XML***”, por lo que

nos dice que hay dos formas de hacerlo:

***BackUp:***

El usuario al cerrar la aplicación, se le da la opción de hacer un backup, por lo que,

si el usuario selecciona que si lo desea, la aplicación crea un archivo “***XML***”, llamado

“***BackUp***”, el cual la próxima vez de que se abra la aplicación, debe leer y cargar en la aplicación. En caso de existir previamente, debe controlar estratégicamente esta solución.

***Share:***

El usuario selecciona la opción de share, por lo que selecciona las frases a compartir

y crea un archivo “***XML***”, llamado “share-fechaYmomentoDeCreacion.xml”, el cual contiene UNICAMENTE las freses seleccionadas por el usuario. Ejemplo: share-dd-mm-aaaa-hhmm- ss.xml (Día, mes, año, horas, minuto y segundo).

Pantalla Principal:

Se espera que el programa principal posea una pantalla como la del siguiente

ejemplo:

La pantalla principal debe:

**(1)** Mostrar TODAS las frases que se han buscado, las cuales son acumulativas (se

pueden hacer 2 búsquedas de 2 frases cada una y al final se deben de mostrar 4

frases diferentes). Debe cargar de “***BackUp***” si la aplicación ya fue antes usada y

el usuario aceptó guardar los datos o aparecer esta sección vacía, pues es la

primera vez que se usa la aplicación o no se desearon guardar los datos.

**(2)** El cliente, además pide que se puedan hacer llamadas repetitivas a la aplicación,

que se puedan realizar desde una petición de una frase hasta 50 frases, las cuales

por el tema del diseño de la API se sabe que pueden incluir frases repetidas, pero

que se deben de manejar.

**(3)** “***Share***” debe dejarme escoger las frases a ingresar en el archivo XML y crearlo

(diseño de dicha pantalla o forma de escoger las frases se deja creatividad del

grupo) y enviar como adjunto al correo electrónico del correo deseado.

**(4)** Debe mostrar la estadística de quien es el personaje con más frases en el sistema,

esto se obtiene con los datos registrados en el diccionario, aquí si el API arrojó

datos repetidos, allí debe considerarse las repeticiones.

**(5)** En el momento de cerrar el programa debe de preguntar si gusta hacer el backup

de las frases actuales ubicadas en la sección (1).

Consideraciones importantes

● Debe considerar el manejo de errores. Por ejemplo, en caso de que no exista conexión

a internet para consultar los datos o la validación de la dirección de correo

electrónico.

● Debe indicar el nombre y dirección URL de las librerías que utilizó a fin de configurar

adecuadamente el ambiente de pruebas para su revisión, en caso de que se requiera.

● En caso de que el archivo ***BackUp.xml*** no exista, debe crearlo en el directorio local.

● Agregue un “***Acerca de...***” para consignar la información de los integrantes del

equipo y llamar desde allí el ***manual de usuario***.

# Scrum

Archivo de Drive

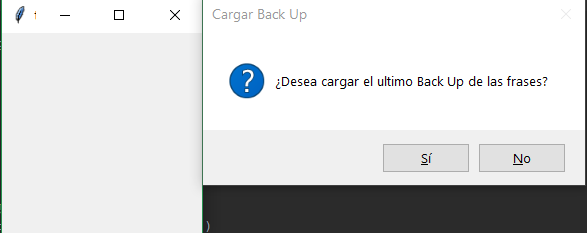
# Estadísticas de tiempo

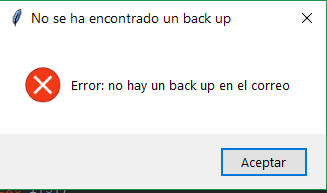
Sebastián López Hererra

Daniel Sequeira Retana

# Manual de Usuario

## Back up de inicio:

Al ejecutar el programa nos preguntará si deseamos cargar in back up del correo, si tenemos uno ya enviado al correo este cargará, de lo contrario indicará que no hay mediante un cuadro de texto. Si presionamos Si cargara el XML con las frases previamente enviadas, si presionamos no simplemente saltara al menú principal

Este será el mensaje indicando que no hay un back up disponible, procederemos a Aceptar y saltaremos a la ventana principal