

# INFORME LABORATORIO 4

Israel Villagra  
16265519-1  
Paradigmas de Programación.

**INDICE**

<b>INTRODUCCIÓN .....</b>	<b>3</b>
<b>PROPÓSITO .....</b>	<b>3</b>
<b>OBJETIVO.....</b>	<b>3</b>
<b>ALCANCE .....</b>	<b>3</b>
<b>DESCRIPCIÓN DEL PROBLEMA .....</b>	<b>3</b>
<b>PROPÓSITO .....</b>	<b>3</b>
<b>ANÁLISIS DEL PROBLEMA .....</b>	<b>3</b>
<b>DISEÑO DE LA SOLUCIÓN.....</b>	<b>4</b>
<b>CONSIDERACIONES DE IMPLEMENTACIÓN .....</b>	<b>5</b>
<b>RESULTADOS OBTENIDOS.....</b>	<b>6</b>
<b>EVALUACIÓN COMPLETA.....</b>	<b>13</b>
<b>CONCLUSIONES .....</b>	<b>14</b>
<b>DOCUMENTACIÓN.....</b>	<b>14</b>
<b>REFERENCIAS .....</b>	<b>14</b>

## INTRODUCCIÓN

### Propósito

El propósito del documento es cumplir con lo exigido en la entrega del laboratorio 4.

### Objetivo

Aplicar lo aprendido del paradigma Orientado a Objetos aplicados en la gestión de eventos.

### Alcance

El documento está redactado según lo requerido en los requerimientos del Informe, por lo tanto, aquello que no aparezca incluido en él se considerará como “fuera del alcance”.

## DESCRIPCIÓN DEL PROBLEMA

### Propósito

El propósito del documento es abordar los requerimientos funcionales que se encuentran informados en el especificados en el documento, imitando (simulando) las funcionalidades de GitHub con un entorno gráfico que permita al usuario manejar los trabajos a través de eventos.

Según lo requerido en el laboratorio 4, se requiere desarrollar una simulación de un sistema de control de versiones de código fuente Git tomando como referencia lo completado en el laboratorio 3, pero esta vez aplicando el manejo de eventos con interfaz gráfica.

## ANÁLISIS DEL PROBLEMA

El simulador de control de código fuente emula una plataforma de control de código fuente GitHub, lo cual consiste en llevar los cambios realizados en los diferentes archivos que compete la solución que se está entregando, pero siendo controlado con zonas de trabajo mientras se van realizando cambios en los archivos indicados. Estos archivos mientras se van realizando cambios que una vez confirmados cambian de carpeta la cual inicia en la zona de trabajo **Workspace**, luego de ser confirmado el cambio pasan a la siguiente zona de trabajo **Index**, una vez en aquella zona de trabajo se copian a la zona de trabajo **Local Repository** lo cual queda registrado en un commit, finalmente al pasar y dejar de manera remota el acceso a los archivos se realiza la copia de la zona de trabajo descrita a la nueva zona de trabajo **Remote Repository**, una vez ya copiado en aquella zona de trabajo se puede obtener la última versión de las modificaciones o cambios que hacia la zona de trabajo **Workspace** como se muestra en la siguiente imagen:

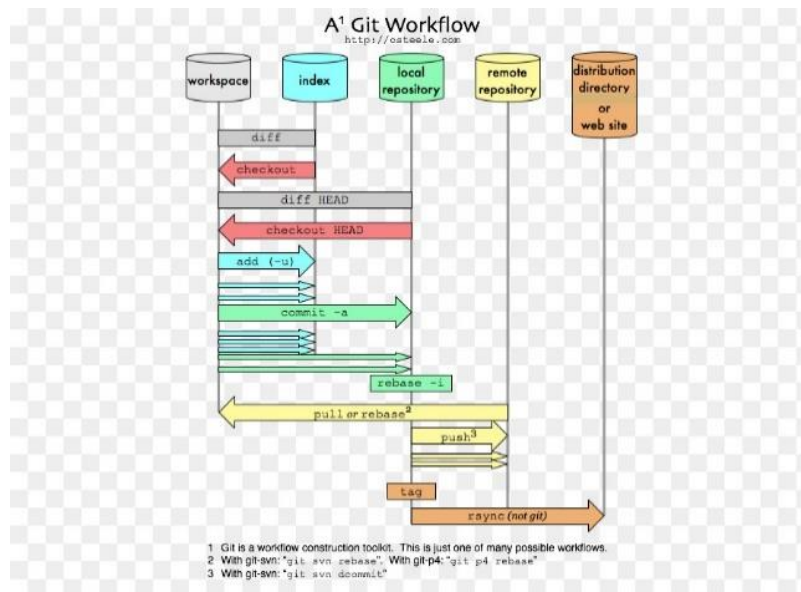


Ilustración 1 Diagrama GitHub

## DISEÑO DE LA SOLUCIÓN

Para abordar el problema se analiza el trabajo realizado por GitHub, por lo que se identifica cada una de las carpetas (Zonas de Trabajo) con la que trabaja la plataforma y su correspondiente trabajo, cuales incorporan archivos o carpetas que en los cuales se han realizado cambio, y una vez confirmados a través de comandos se copian a la siguiente carpeta, llegando finalmente a la carpeta Remote Repository, el cual es el destino final.

Por lo cual, se define las clases que, emulan las carpetas (Zonas de Trabajo) existentes en GitHub, además varias clases que contienen la dirección de subida de archivos a su siguiente nivel, en donde se especifica desde que carpeta proviene la siguiente

Se crean una clase Repositorio que la cual contiene una lista de Zonas de Trabajo, las cuales contienen lista de archivos de texto plano y un arreglo de los commits que se han desarrollado en aquella zona de trabajo.

Para incorporar la solución del cual exige una interfaz de usuario gráfica, se han creados 3 soluciones, las cuales son:

- Laboratorio4.Controller
- Laboratorio4.Entities
- Laboratorio4.WindowsForm

### Laboratorio4.Controller

Contiene una clase toda la lógica de la copia de datos desde una zona de trabajo a la siguiente.

### Laboratorio4.Entities

Contiene la Lista de entidades que son utilizadas con la solución con el paradigma orientado a objetos.

### Laboratorio4.WindowsForm

Contiene todos los formularios y validaciones para enviar al controlador que contiene la lógica de las zonas de trabajo.

Todo lo mencionado anteriormente fue realizado por petición por los requerimientos obligatorios.

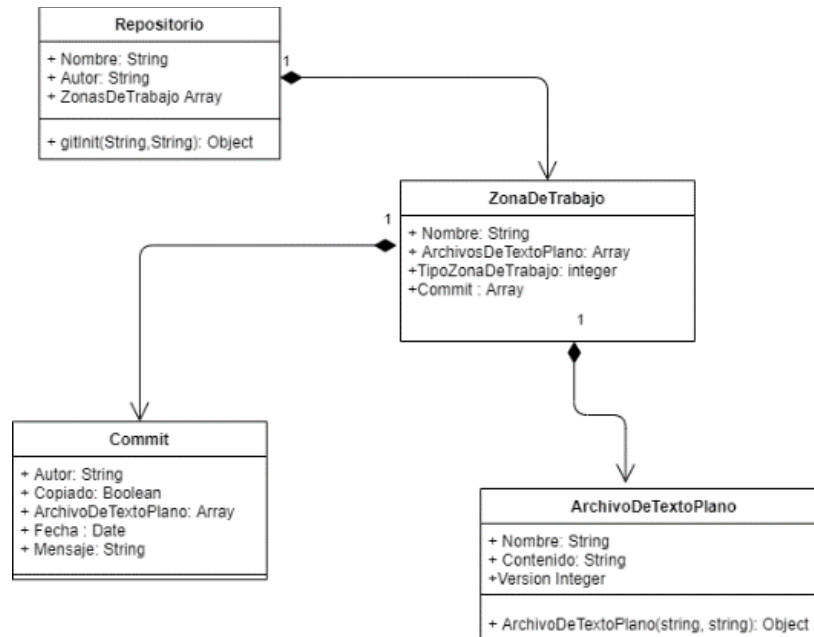
### CONSIDERACIONES DE IMPLEMENTACIÓN

Se fue considerado la implementación de las clases, por el paradigma en el cual trabajar se ha requerido, sin movimientos de archivos físicos, sólo de manera de listas, pero con la integración de los manejos de eventos con la solución gráfica.

La estructura del proyecto fue realizada según sus requerimientos obligatorios, al igual que el compilador utilizando Visual Studio Professional 2015, con sus bibliotecas incorporadas que pertenecen al compilador.

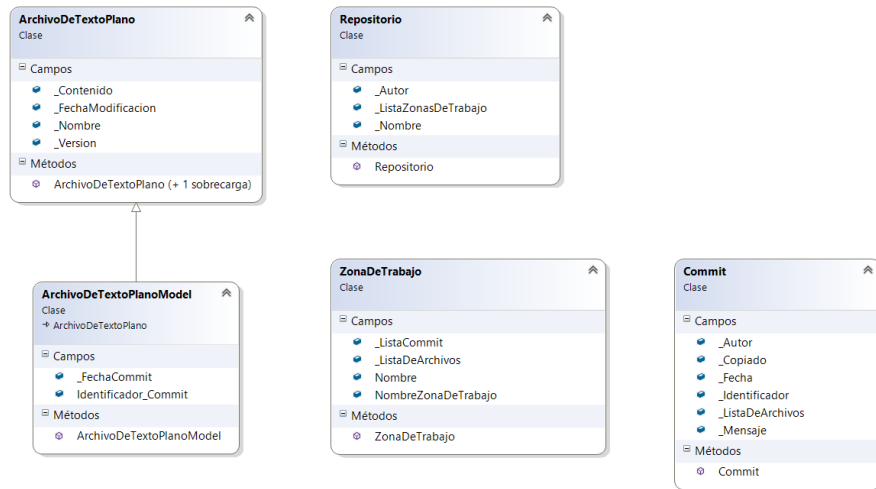
### 6.-Requerimiento Funcional No Obligatorio

Se incluyo solamente la versión del archivo de texto Plano.



### Diagrama de Clases de la Solución

Si bien no se encuentran relacionadas de manera directa, se debe a que no está implementado con relaciones de bases de datos.



### RESULTADOS OBTENIDOS

Se realiza la obtención de la información de los predicados que son solicitados de manera obligatoria

Se debe establecer como proyecto de inicio el Laboratorio4.WindowsForm.

Se puede considerar de igual manera estos puntos que serán abordados como apoyo al manual de usuario

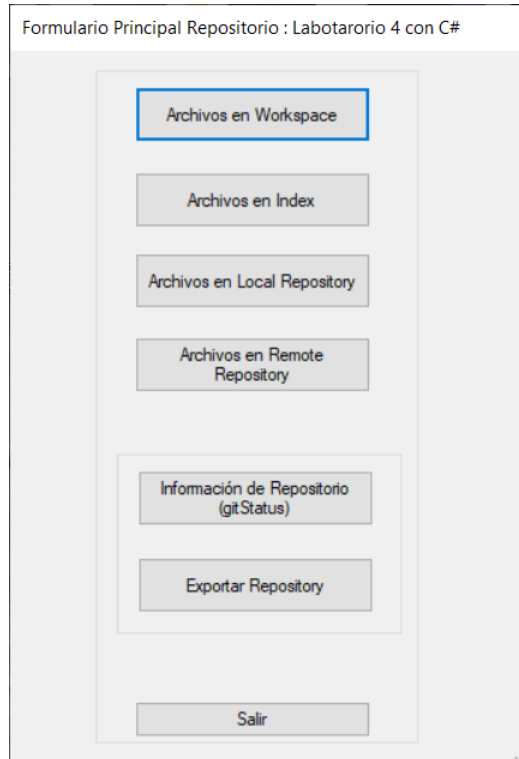
- **gitlnit:**

En el formulario de debe ingresar el nombre del autor del Repositorio y el nombre.

Una vez presionado el botón “Aceptar” aparecerá la siguiente imagen, titulado con el nombre del laboratorio ingresado.

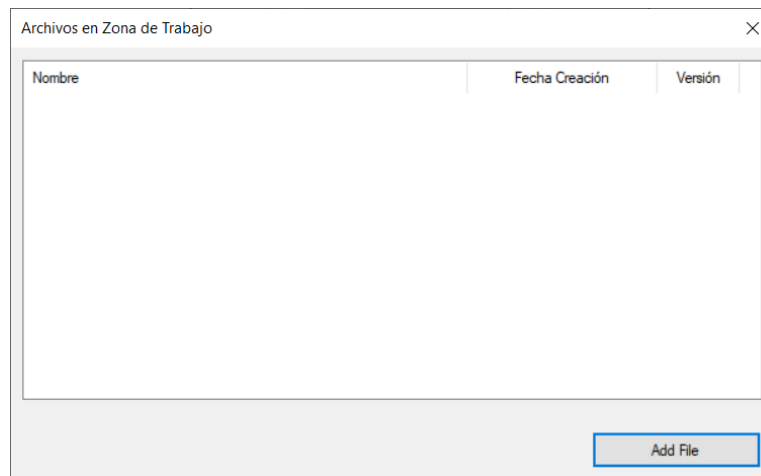
Nota: Si necesita más información puede presionar el botón “Información del Repositorio (GitStatus)”

Formulario Principal Repositorio : Labotatorio 4 con C#



- **gitAdd:**

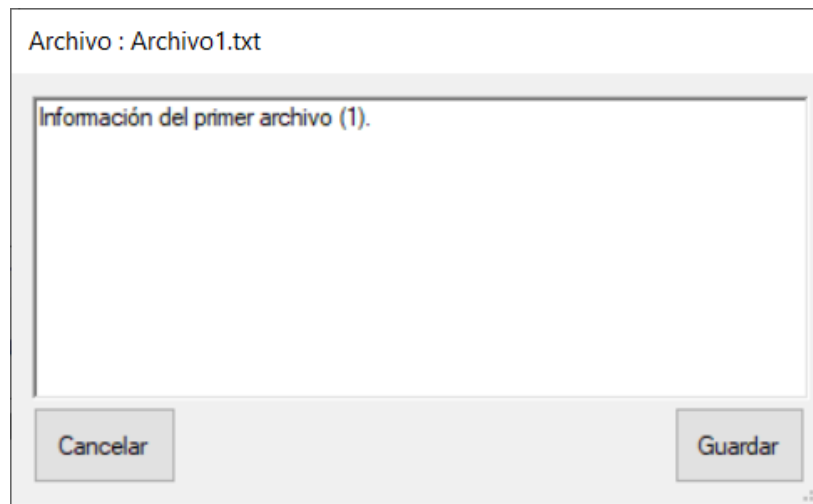
Se debe presionar el botón “Archivos de WorkSpace”, la cual desplegara un nuevo formulario, en donde debe presionar el botón “Add File”



Presionado el botón el sistema le indicará que ingrese el nombre del archivo.



Una vez presionado el botón aceptar podrá ingresar el contenido del archivo como se muestra en la siguiente imagen:



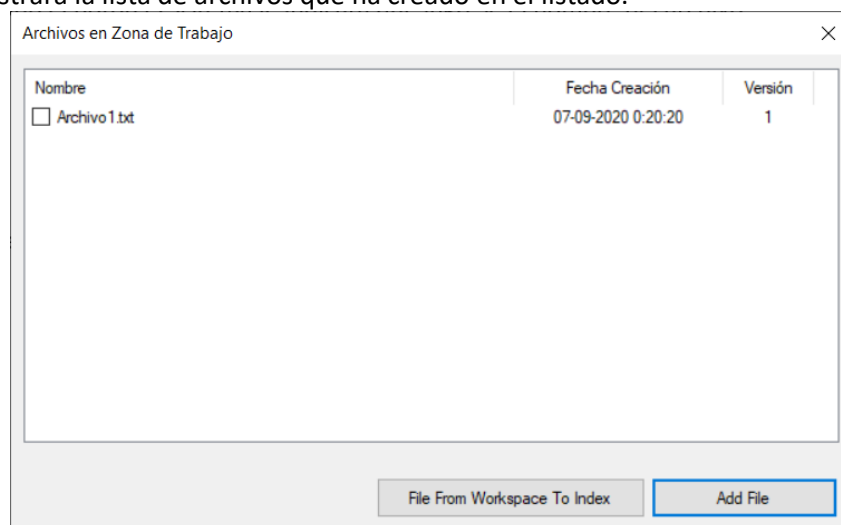
Archivo : Archivo1.txt

Información del primer archivo (1).

Cancelar Guardar

Una vez ingresada la información presionar el botón “*Guardar*”, de caso contrario si no es necesario ingresar el nuevo archivo, presionar el botón “*Cancelar*”

El sistema mostrará la lista de archivos que ha creado en el listado.



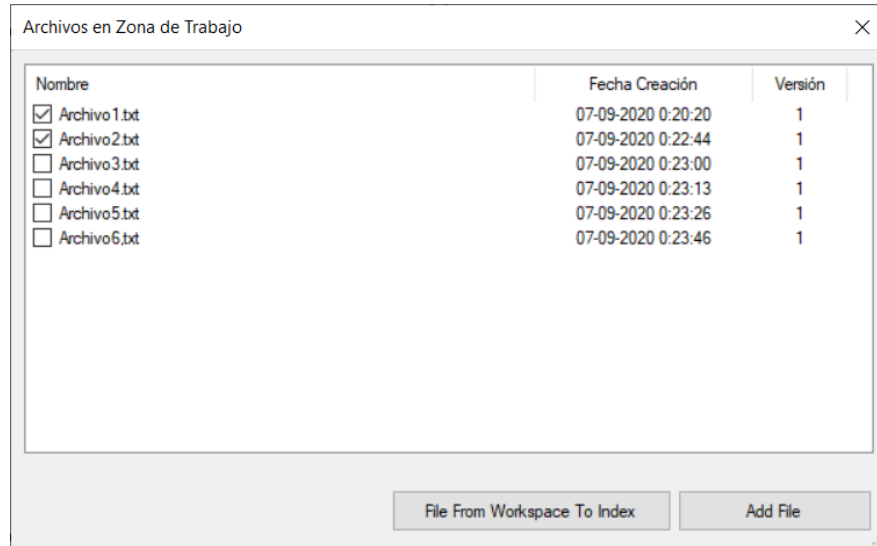
Nombre	Fecha Creación	Versión
<input type="checkbox"/> Archivo1.txt	07-09-2020 0:20:20	1

File From Workspace To Index Add File



- **gitCommit:**

Se debe seleccionar los archivos que se debe realizar el gitcommit y presionar el botón “*File From Workspace To Index*”.



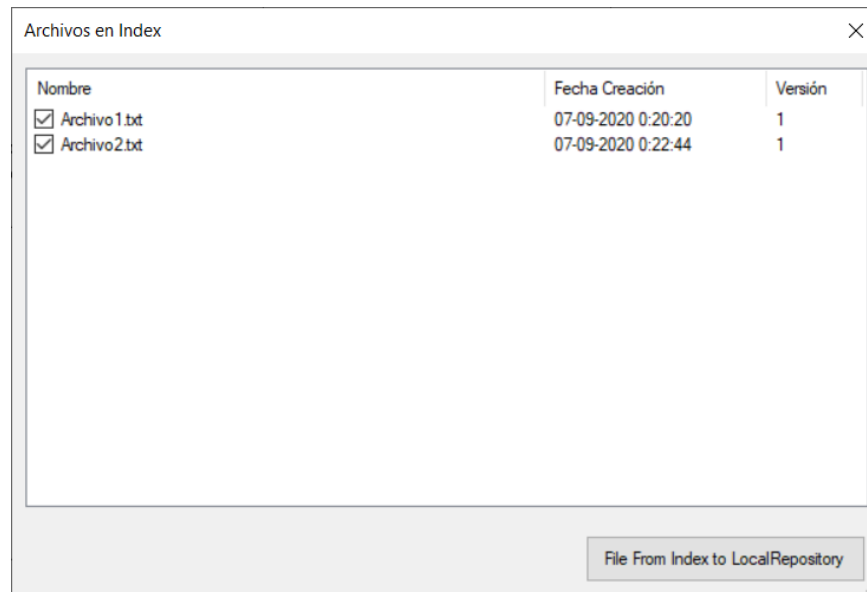
El sistema le entregará el resultado de la operación.



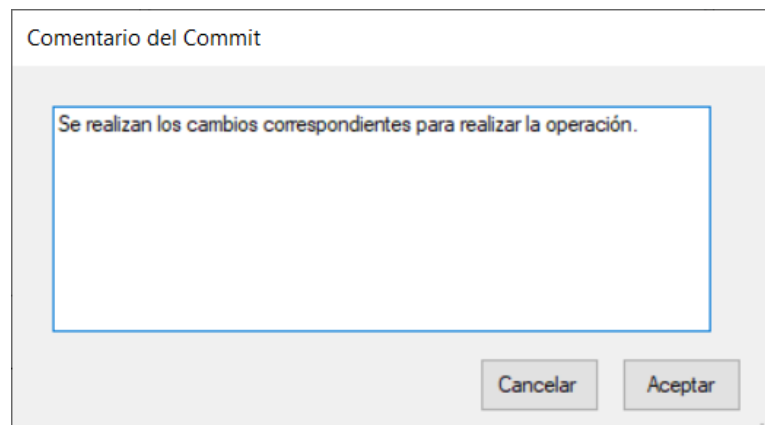
Para poder verificar puede cerrar el formulario, por lo que aparecerá el formulario Principal, y debe presionar el botón “*Archivos en Index*”.

- **Commits:**

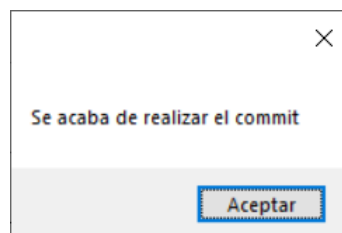
Ingresando al formulario que despliega todos presionando el botón “Archivos en Index” en el formulario principal, aparecerá la lista de archivos, de los cuales debe seleccionar los que se realizarán el commit, por lo que debe presionar el botón “File From Index to Local Repository”.



El sistema le pedirá que ingrese la información relevante al evento de copiar los archivos para copiar los archivos a la siguiente Zona de Trabajo Local Repository.

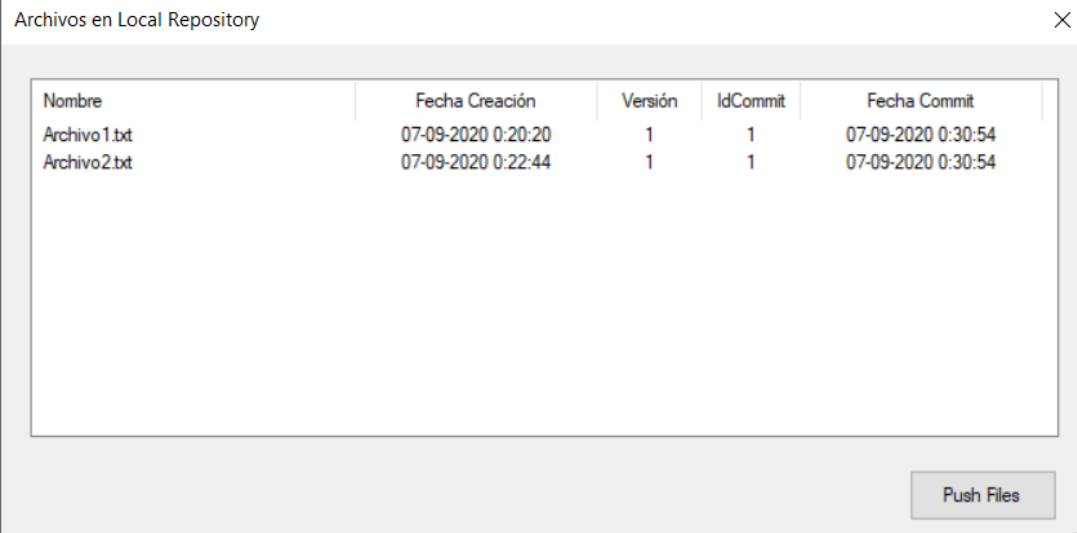


El sistema le desplegará un mensaje con el resultado de la operación



- **Push**

Para realizar el proceso debe ingresar al formulario principal, y presionar el botón “*Archivo en Local Repository*”, una vez desplegado el formulario debe presionar el botón “*Push Files*”



Nombre	Fecha Creación	Versión	IdCommit	Fecha Commit
Archivo1.txt	07-09-2020 0:20:20	1	1	07-09-2020 0:30:54
Archivo2.txt	07-09-2020 0:22:44	1	1	07-09-2020 0:30:54

Push Files

Una vez finalizado el proceso el sistema le desplegará un mensaje con el resultado de la operación

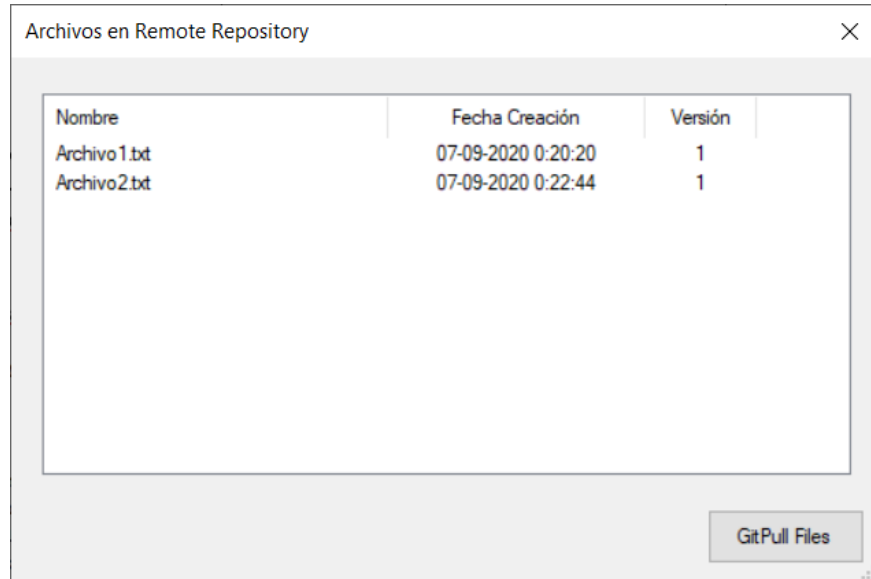


Se han copiado los archivos en Remote Repository

Aceptar

- **Pull:**

Para realizar el trabajo de debe ingresar al panel de principal y presionar el botón “Archivos en Remote Repository”, quien desplegará el siguiente formulario.

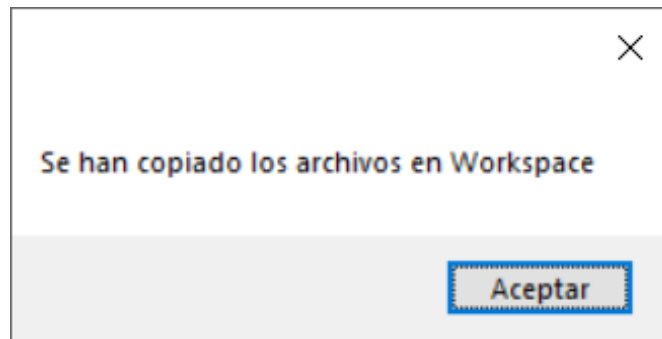


Nombre	Fecha Creación	Versión
Archivo1.txt	07-09-2020 0:20:20	1
Archivo2.txt	07-09-2020 0:22:44	1

GitPull Files

Para realizar la operación de debe presionar el botón “Git Pull Files”

El sistema le desplegará un mensaje con el resultado de la operación.



Se han copiado los archivos en Workspace

Aceptar

- **gitStatus**

Para visualizar el estado de debe presionar el botón “*Información del Repositorio (gitStatus)*”, quien desplegará la información correspondiente al estado de cada una de las zonas de trabajo

The screenshot shows the 'GitStatus' application window with the following sections:

- Información del Repositorio**
  - Nombre : Labotarorio 4 con C#
  - Autor : Israel Villagra
- WorkSpace**
  - Cantidad Archivos : 6

Nombre Archivo	Versión
Archivo3.txt	1
Archivo4.txt	1
Archivo5.txt	1
Archivo6.txt	1
Archivo1.txt	1
Archivo2.txt	1
- Index**
  - Cantidad Archivos : 2

Nombre Archivo	Versión
Archivo1.txt	1
Archivo2.txt	1
- Local Repository**
  - Cantidad Commits : 1

Id	Fecha	Cantidad Archivos
1	07-09-2020 0:30...	2
- Remote Repository**
  - Archivos Iguales : No

Nombre Archivo	Versión	Fecha	Mensaje
Archivo1.txt	1	07-09-2020 0:20:20	Archivos Iguales
Archivo2.txt	1	07-09-2020 0:22:44	Archivos Iguales
Archivo3.txt	1	07-09-2020 0:23:00	Archivo No Existe en Remote Repository
Archivo4.txt	1	07-09-2020 0:23:13	Archivo No Existe en Remote Repository
Archivo5.txt	1	07-09-2020 0:23:26	Archivo No Existe en Remote Repository
Archivo6.txt	1	07-09-2020 0:23:46	Archivo No Existe en Remote Repository

Si bien no es requerimiento, se ha agregado el formulario una comparación de los archivos en Remote Repository con el WorkSpace indiferente si existen la misma cantidad de archivos, por lo que cuando se realiza la operación se compara archivo por archivo ya sea por Remote Repository como Workspace para conocer que archivos existen en Workspace con Remote Repository.

## EVALUACIÓN COMPLETA

El sistema está orientado a satisfacer lo requerido obligatorio.

Si bien ha incorporado en el menú principal una pestaña en donde se puede exportar por XML el repositorio no se contempla en los requerimientos o funciones extras, pero fue incluido para poder realizar las pruebas con los avances del desarrollo de la aplicación, el cual, de adjunta, que es el mismo que se ha mostrado en el ejemplo. “Labotarorio 4 con C#.xml”.

Se ha incorporado otro archivo exportado, pero con la diferencia de un archivo con diferente versión” *Labotarorio 4 con C# V2.xml*”.

Si bien se realizó la evaluación completa de los requerimientos obligatorios, utilizando el menú de usuario según lo requerido que puede ser verificado con el código fuente compartido y copiado en el archivo adjunto.

## CONCLUSIONES

El paradigma Orientado a Objetos otorga una nueva forma de pensar cuando se interactúan muchas entidades en el proceso de negocio, permitiendo diagramar con abstracción la solución a encontrar del problema o automatización.

En conjunto con los diagramas UML, entrega una herramienta para acotar los requerimientos, lo cual permite desplegar a través de diagramas los servicios o procesos que realizará la solución y acotar el alcance para que el usuario normal sin especialización lo pueda comprender.

En cuanto el Framework que incorpora C# contiene una gran cantidad de paradigmas con los que se puede trabajar, incorporando un buen soporte para la realización de soluciones que necesiten múltiples paradigmas.

## Documentación

Se realiza la documentación del código según lo requerido por **XML (link)**, indicador en los requerimientos

## Referencias

- <https://docs.microsoft.com/en-us/dotnet/csharp/codedoc>