DIARIO DI BORDO – progetto DSB

1. 20.9-1.10

Incontro con Michele:

discussione preliminare sul progetto: realizzazione GUI che andrà a sostituire quella attualmente in uso per gli esperimenti del macchinario la quale risulta funzionale unicamente se l’utilizzatore è lo sviluppatore della stessa, C#, Visual Studio 2019 Enterprise.

Visti gli use cases. Discussione sulla scelta fra WPF, Winforms, o eventuali altre librerie grafiche.

Sottoscritto:

Lettura documentazione su C# (*Programmare con C#8 | Guida Completa*, Antonio Pelleriti, ISBN\_978-88-6895-769-8), realizzazione Hello World grafico con C# e WPF. Realizzazione primi mockup (in seguito risultati troppo graficamente indipendenti fra loro, nel senso che ogni mockup era una finestra separata dalle altre; ma poiché erano stati inizialmente preparati sulla base del singolo use case anziché accorpati in un’unica interfaccia grafica).

1. 1.10-8.10

Incontro:  
il prodotto dovrà gestire la parte offline, ossia visione dei dati salvati sul file system delle precedenti deposizioni, osservare gli attuali setup dei dispositivi associati a un dato esperimento/deposizione; e la parte online, ossia la creazione di un nuovo esperimento con i relativi dispositivi e setup associati.

Il setup dei dispositivi è delegato alle applicazioni dedicate ai dispositivi.

Possibilità di usare il database anziché il file system?

Completare la demo Hello World con una simulazione di errore (con message on top, principale inconveniente riscontrato da Michele per quanto concerne Winforms).

Sistemare i mockup.

Sottoscritto:

Implementazione Thread nell’Hello World: il messaggio appare on top in momento relativamente casuali. Sistemazione dei mockup. Continuazione della lettura della documentazione su C#.

1. 8.10-15.10

Incontro:  
decisione finale sull’uso della libreria grafica: WPF. In .NET 5 e.NET 6 è ancora presente, inoltre è stato valutato che l’applicazione rimarrà dedicata all’ambiente desktop; difficilmente avrà sviluppo in ambiente mobile.  


Fonte: <https://www.winfxitalia.com/articoli/dotnet6/introduzione-dotnet6-csharp10.aspx>

1. 15.10-22.10

Incontro:

1. - deep dive in wpf (documentazione e video) -> cosa va in XAML, cosa in controller, ecc
2. - completare un primo scheletro del mockup in WPF
3. - sistemare mockup:

* led status da togliere (i bottoni start e feeder on possono già indicare lo status),
* aggiungere GCODE e DSB software nella parte status,
* sugg: l'aggiunta di una tab (segno +) potrebbe copiarmi nella nuova tab un esperimento attualmente aperto in modo da eseguire dei confronti,
* un'immagine sola nei record e no ingrandimento,
* separare mockup nuovi e vecchi in due documenti

     4.- focus di coerenza: mostrare i bottoni di experiment o deposition quando essi sono selezionati, altrimenti no (mostrare le informazioni necessarie all'attuale focus).

!!Sono stati passati un esperimento con delle deposizioni per i test!!

Sottoscritto:  
 inizio implementazione dell’applicativo basato sul mockup.

1. 22.10-29.10  
     
   Incontro:

**Vacanze morti  
Mariano assente/limitato per il progetto: corso di formazione 1-6 novembre "full immersion"!**

Mockup  
Start --> automatic start (sniff start)  
Feeder On --> MANUAL start  
--> ambedue spostabili a seconda di dove sono comodi e smart!

Use case  
compress experiment (dapprima comprime tutte le deposizioni singolarmente e poi l'esperimento).

Editor immagini  
Michele: progetti ingegneria del software 1 --> editor di immagini  
Mariano: guardare il progetto di imagik di De Sanctis, **per ispirazione**  
Verificare: Open source wpf image viewer

Image viewer  
Setting come contrasto automatico, etc, o embedded o tramite xnviewer (pulsante, mouse destro, etc)  
Preview delle immagini embedded --> pulsante di ordine per immagini più potenti o altri filtri.  
Utile all'utente per verificare che il setup della camera sia corretto senza scorrere tutte le immagini  
  
Immagine di due camere diverse affiancate

Apertura setup device  
SensorTools v1.18  
IDS software suite (Melt Pool)  
Text CN (già in possesso Mariano)  
DSB software --> selezionare un file  
Smartekvision\_CameraSuite\_v3

Sottoscritto:

Cercati alcuni image viewer in WPF, ma non sono stati implementati

1. 29.10-12.11

Incontro:

Directory  
Possibilità di directory: aprire la directory dell'esperimento selezionato o aprire la directory degli esperimenti.

Tab esperimenti  
Non aprire tab identiche dallo stesso experiment item!!

Demo finale  
Immagini live (Smartek) e immagini catturate/registrate precedentemente (vedi quelle nelle deposizioni)

XAML  
Integrazione file xaml esterni (lavorarci su un esempio esistente e piccolo)

Ricevuto  
Da Michele: setup di smartek e nuova deposizione rar

Sottoscritto:  
corso G+S calcio bambini dal 1.11 al 6.11 durante il quale ho potuto mettere poco le mani sul progetto. Test tramite gli user control nell’Hello World: funzionante! Lo user control permette di separare fra loro diverse strutture WPF costruite in file xaml separati. È sufficiente istanziare il cs degli user control.

Settimana di infortunio dal 8.11. Sono riuscito ad aprire le tab per gli esperimenti e navigare fra le immagini della deposizione aperta dall’esperimento stesso.

Bottoni dell’apertura delle directory non ancora implementati. Progettati e scritti alcuni user control.

Cambio mockup e implementazione: si apriva una singola deposizione per ogni esperimento, anzi una deposizione sola in totale, quindi l’applicativo è stato ridisegnato affinché ogni tab degli esperimenti abbia un tabcontrol dedicato alle deposizioni.

1. 12.11-19.11  
     
   Incontro:  
     
   Mariano

1-Diario sul percorso: ogni 2-4 h un commento da mettere in github o in un text file (x non perderselo)

2-Creare le issue per le cose da fare

3-Mostrare le immagini e slider (possibilità d'inserire un numero di millisecondo)

4-Trovare un modo di gestire visione di dettaglio e

globale

5-Visione di dettaglio, basata sul millisecondi, e

6-visualizza l'immagine, LaserOn, PowerFeedback e

temperatura

7-Visione globale: aprire un editor notepad++ da un

explorer di files del folder della deposition

Nota

* I protocolli del CN possono essere piu' di uno per deposizione
* Online: create experiment => DB dependencies
* visualizza l'immagine MP, CamEsterna, LaserOn, PowerFeedback e temperatura



Sottoscritto:

15.11 🡪 creazione di issues e milestones su github, creazione di un file Microsoft Projects con le deadlines.

Bug: errore nello spostamento dei TabItem degli esperimenti (nullPointerException) 🡪 depoTabControlController.setPath((string)tabItem.Header);

🡪risolto il 15.11:  
Fixed:

TabItem tabItem = (TabItem)mainTabControl.SelectedItem;

if(tabItem != null)

{

depoTabControlController.setPath((string)tabItem.Header);

}

16.11 🡪

file explorer trovato al link:

* <https://github.com/mikependon/RepoDB.Tutorials>
* <https://github.com/AngryCarrot789/FileExplorer> 🡪 … insomma
* <https://www.youtube.com/watch?v=aonRoEokQeY> 🡪 soluzione usata



1. 19.11-26.11  
   Incontro:  
     
   0) *Adattare il tuo file system al nuovo formato: pf inserire la foto in diario di bordo*

1*) Xn view ti dà il mio stesso errore, on next image crasha?*

2) Cliccando sull'immagine a destra nel file explorer, carica quella immagine a sinistra

3) *Close Tabs non fa nulla: ok => Mettilo fra parentesi (Close Tabs)*

*Fare cosi dappertutto e mantenerlo*

4) Collegare il commento dell'experiment al file dell'experiment, che c'è sempre, ev. vuoto

5) *Usare i ms della prima e ultima immagine per estrarre i ms significativi e usare quelli per tutto il displaying dei ms (immagini e dati ecc)*

6) Nei dati inserire: temperatura, LaserOn, PowerFeedback,

Visualizzare il dato relativo al ms esatto, oppure una media degli ultimi 3 valori piu' vicini

7) Poter inserire un numero di millisecondo e visualizzare i dati relativi

8) Header del file explorer fondato su web browser: per ora lasciarlo cosi

Problematica: è molto lento

9) Il file .mpp è bello ma lo integriamo completamente in github>https://github.com/MarianoFasano/DSB/milestones

Nota: Le milestones sono sotto le issues!

Abituiamoci ad avere tutto nel kanban: https://github.com/MarianoFasano/DSB/projects/1

Agganciare OGNI ISSUE ad una milestone, così abbiamo il controllo della rilevanza agganciato alla priorità (scadenza)

10) Settimana prox abbiamo il MS1

Sottoscritto:

21.11: adattato il file system, sia sul fisso sia sull’hard disk con i dati usati sul portatile.

XnView dà lo stesso problema che riscontra Michele.

Il titolo dei tasti non ancora implementati è stato messo fra parentesi.

Le immagini, così come i dati del pirometro e del CN, non sono consecutivi di ms in ms, spesso e volentieri saltano di qualche ms.

22.11: usare sempre lo stesso oggetto, senza istanziarne dei nuovi ogni volta, per myDepoData. Assegnare di volta in volta il valore del dictionary all’istanza già esistente sovrascrivendola (SSD riscrive tutto il blocco di memoria…)

Mostrare un messaggio quando i ms selezionati non coincidono con quelli dell’immagine? Meglio avere i dati che non collimano con i ms passati dell’utente o meglio non avere l’immagine (che salta subito all’occhio!).

23.11: implementare un binary search a seconda della lunghezza del valore passato al textbox!?

Rinominato initButtonsAction in initControlsAction in DepoTabControlController, poiché non si assegnano unicamente i bottoni ma anche textbox e labels (ciò che serve).

Search textbox 🡪 aggiunto metodo, non implementato, quando si preme enter this.searchMs.KeyDown += SearchMs\_KeyDown;

24.11: implementato un binary search per ms “piccoli” (ossia che vanno da 0 al valore massimo riportato dalla label) 🡪 durante la fase di refactoring valutare un pattern per il dispatcher!

Implementata anche la versione con i ms “lunghi”.

Ricerca su come leggere un file di testo, saltarne le righe, ecc…

Leggo ogni riga che viene salvata in un IEnumerable castata in seguito a List da me. Rimuovo in modo hard le righe che non mi interessano e dopodiché eseguo la binary search, al momento sui ms in versione lunga. Il risultato è corretto, ma ve ne possono essere più di uno. Bisogna capire se fare la media o che cosa, ma al momento qualcosa ritorna.

Probabilmente per il CN non possiedo il file corretto, manca laserOn e powerfeedback

Esempio di binarySearch con numeri lunghi.

private string pyroLongBS(List<string> pyroLines, long searchedMs, int left, int right, string min)

{

count++;

if (left > right)

{

//Do nothing!

}

int middle = (left + right) / 2;

string originalElement = pyroLines.ElementAt(middle);

string element = extractFromPyroLine(originalElement);

if (count > 25)

{

resetCounter();

return pyroLongBS(pyroLines, searchedMs - 1, 0, pyroLines.Count() - 1, min);

}

if ((long.Parse(element) - long.Parse(min) == searchedMs))

{

return originalElement;

}

else if ((long.Parse(element) - long.Parse(min) > searchedMs))

{

return pyroLongBS(pyroLines, searchedMs, left, middle - 1, min);

}

else

{

return pyroLongBS(pyroLines, searchedMs, middle + 1, right, min);

}

}

Cercare di capire come fare l’update delle label.

Popolato il file CN con dati randomici per quanto concerne LaserOn e PowerFeedback (programma in C#).

Inserito un extracounter per aumentare il valore cercato dopo un certo numero di tentativi, questo poiché si presentavano eccezione (ad es. cercando il valore 5). Tolto in seguito poiché il problema era il valore che diveniva negativo (dopo 20 tentativi, bigO, della binarysearch si diminuiva il valore cercato di 1) poiché per valori piccoli spesso non veniva trovato alcun riscontro. Conseguentemente il controllo di condizione è stato cambiato anziché sul secondo counter sul

26.11: update label risolto 🡪 cambiato in TextBox readonly!!

1. 26.11 – 3.12  
     
   Incontro:  
     
   Nota per Michele:

Directory aggiornata per far girare l'attuale DSB di Mariano:

D:\\_GC\_DataSetBuilder\DBItems\gc\_0\_4\_7\_prod\_gcmp\Experiments\Experiment\_2021\_9\_14\_\_11\_13\_42\Deposition\_2021\_09\_14\_\_11\_37\_42

ok 0) Adattare il tuo file system al nuovo formato: pf inserire la foto in diario di bordo

si 1) Xn view ti dà il mio stesso errore, on next image crasha?

x 2) Cliccando sull'immagine a destra nel file explorer, carica quella immagine a sinistra

ok 3) Close Tabs non fa nulla: ok => Mettilo fra parentesi (Close Tabs)

Fare cosi dappertutto e mantenerlo

x 4) Collegare il commento dell'experiment al file dell'experiment, che c'è sempre, ev. vuoto

ok 5) Usare i ms della prima e ultima immagine per estrarre i ms significativi e usare quelli per tutto il displaying dei ms (immagini e dati ecc)

x 6) Nei dati inserire: temperatura, LaserOn, PowerFeedback,

Visualizzare il dato relativo al ms esatto, oppure una media degli ultimi 3 valori piu' vicini

x 7) Poter inserire un numero di millisecondo e visualizzare i dati relativi

Bug su inserimento diretto del ms, es. 1631614771990

x 8) Header del file explorer fondato su web browser: per ora lasciarlo cosi

Problematica: è molto lento

ok 9) Il file .mpp è bello ma lo integriamo completamente in github>https://github.com/MarianoFasano/DSB/milestones

Nota: Le milestones sono sotto le issues!

Abituiamoci ad avere tutto nel kanban: https://github.com/MarianoFasano/DSB/projects/1

Agganciare OGNI ISSUE ad una milestone, cosi abbiamo il controllo della rilevanza agganciato alla priorita (scadenza)

10) Settimana prox abbiamo il MS1

x Gestire sia depo folder con vecchia struttura che nuova

x Visualizzazione dati: vederli sin dall'inizio, settati sul ms 0

x Gestire se c'è o meno il dato laser on e power feedback

x Togliere la data alla deviceImaging, e visionBox pure, poichè le date le inserisco all'interno di questo folder

x Chiudere experiment tab

x Gestire il caso in cui manca un file, es pirometro

x Visualizzazione dei ms opzionale come flag nella GUI:

VerA)

ms1631612262574

ms1631618162652

VerB)

12262574

18162652

x Idea: tenere l'ultimo experiment aperto, e ultima depo aperta

Meeting di venerdi 17.12 spostato alle 14:30 (alle 16:30 noi ARM andiamo a festeggiare il Natale)

Meeting di venerdi 24.12 spostato a merc 22.12 1730

Next meeting: venerdi 3.12, con demo ad Ambra e Stefano

Sottoscritto:

28.11.2021

* OK Togliere la data alla deviceImaging, e visionBox pure, poichè le date le inserisco all'interno di questo folder
* OK Visualizzazione dati: vederli sin dall'inizio, settati sul ms 0
* OK Visualizzazione dei ms opzionale come flag nella GUI:  
  VerA)  
  ms1631612262574  
  ms1631618162652  
  VerB)  
  12262574  
  18162652
* OK Gestire sia depo folder con vecchia struttura che nuova
* OK Gestire se c'è o meno il dato laser on e power feedback
* OK Gestire il caso in cui manca un file, es pirometro
* Chiudere experiment tab
* Idea: tenere l'ultimo experiment aperto, e ultima depo aperta

Cartelle rinominate, sono stati tolti i dati relativi alla data di creazione della deposizione.

Creazione di due funzioni che racchiudono le due ricerche (ms “corti” e ms “lunghi”).

All’inizializzazione della prima immagine viene lanciata una ricerca che cerca i dati relativi alla prima immagine.

30.11:

“Fixed” il System.StackOverFlow dovuto alla ricerca del valore 1631614771990 inserendo un counter, un controllo fintanto che non arriva al 100, quando avviene l’offset viene moltiplicato di 7.

1.12:

Aggiunto il CheckBox per mostrare o non mostrare la versione estesa dei ms.

Sistemata meglio la funzione che gestisce la larghezza della colonna degli esperimenti al click del bottone: si allarga alla larghezza automatica iniziale e collassa a zero.

Gestione nel caso non fossero presenti dei file, es pirometro. Condizione di controllo: if(myDepoData.getPyrometerList().Any()).

Gestione nel caso non fossero presenti i dati di laserOn e powerFeedback (nelle deposizioni più vecchie difatti non sono presenti) nel file CN. Soluzione portata:

if (!laserOn.Any())

{

laserOn = "No value";

}

else

{

index = laserOn.IndexOf("\t");

laserOn = laserOn.Substring(0, index);

}

return laserOn;

if (!powerFeedback.Any())

{

powerFeedback = "No value";

}

return powerFeedback;

Si verifica se la stringa contenente i dati contenga effettivamente qualcosa, in caso contrario sarebbe vuota.

2.12:

Gestione della vecchia struttura del file system. Se la directory contiene dei files si procede alla vecchia maniera, mente se non vi sono file (solo folder) si procede con la nuova struttura. Condizioni if() ed else if().

Sito da controllare per chiudere la tab:

<https://www.codeproject.com/Articles/493538/Add-Remove-Tabs-Dynamically-in-WPF>

<https://stackoverflow.com/questions/3468866/tabcontrol-with-add-new-tab-button>

1. 3.12-10.12

Incontro:

1. OK Togliere la data alla deviceImaging, e visionBox pure, poichè le date le inserisco all'interno di questo folder
2. OK Visualizzazione dati: vederli sin dall'inizio, settati sul ms 0
3. OK Visualizzazione dei ms opzionale come flag nella GUI:  
   VerA)  
   ms1631612262574  
   ms1631618162652  
   VerB)  
   12262574  
   18162652
4. ToDo Gestire sia depo folder con vecchia struttura che nuova
5. OK Gestire se c'è o meno il dato laser on e power feedback
6. OK Gestire il caso in cui manca un file, es pirometro
7. ToDo Chiudere experiment tab
8. ToDo Ricordare il path dell'ultimo experiment aperto, e ultima depo aperta  
   => Aggiungere un menu all' experiment, e uno alla depo, e salvare nel file .ini l'informazione  
   => Guardare come si fa in visual studio: non c'è da inventare nulla!

New items:

1. URGENTE E RILEVANTE:
   1. Mettere commenti nel codice, per il tuo povero docente 😊
   2. Ogni costante DEVE essere commentata!
   3. PF avvisami quando hai fatto un primo giro (non venerdi alle 15)
2. Evitare linee lunghe di codice, come questa ma creare variabili locali, piu' facili da debuggare:  
   BitmapImage bitmapImage = new BitmapImage(new Uri(dataPath + @"\" + myDepoData.getImageDirectory() + myDepoData.getImages().ElementAt((int)myDepoData.getActualImage()), UriKind.RelativeOrAbsolute));
3. ~Add a tool bar, con icona refresh per aggiornare il file system (chiude l'app e la restart con il file system aggiornato), e reset layout e close tabs
4. OK Punto 3 dell'ultima volta: mostriamo e gestiamo solo i ms assoluti
5. OK App si apre in formato ridotto, usare this.WindowState = FormWindowState.Maximized;
6. Il tuo ordinamento degli experiment è diverso rispetto a quello del file system: prova a vedere se ci sono opzioni d'ordinamento
7. App dà errore su Experiment\_2021\_10\_22\_\_10\_37\_16, non appena si cerca di aprirlo (roba di path e collection vuota prob)
8. OK Binary search slow: va fatto sulla lista (carica tutti i valori all'inizio) e non facendo file access tutte le volte!  
   List<string> pyroLines = File.ReadAllLines(myDepoData.getPyroFileDirectory() + @"\" + myDepoData.getPyrometerList()[0]).Cast<string>().ToList();
9. OK Coesione: DepoTabControlController va suddiviso in sotto parti
10. Stefano Ambra Michele feedback:
    1. Scorribilità degli experiment items
    2. Ricercare per file di setup: vedere con Banfi
    3. Inserire una foto del provino sotto alla lista degli experiments
    4. Signal plotter: <https://nugetmusthaves.com/Package/ScottPlot.WPF>

|  |  |
| --- | --- |
|  | [ScottPlot.WPF - NuGet Must Haves Package](https://nugetmusthaves.com/Package/ScottPlot.WPF)  [nugetmusthaves.com](http://nugetmusthaves.com/)  This package provides a WPF control to interactively display ScottPlot graphs in WPF applications. This package targets both .NET Framework and .NET Core. |

* 1. Dividere DSB parte offline dalla parte online: Banfi stuff

5.12:  
Refactoring della classe DepoTabControlController (ricerca divisa in tre altre classi, NON ancora terminata).

6.12:

Lettura del capitolo di LINQ da libro di testo e visione di alcuni video esplicativi (2x, 5-6min).

Prima aggiunta di commenti al codice, mancano ancora alcune classi.

7.12:

Le ricerche dei valori sono da separare in singole funzioni di ricerca: problema di ridondanza nel caso di navigazione delle immagini tramite i bottoni di back/next.

Limitare la ricerca ai valori lunghi di ms, assoluti. L’altro va tolto per facilità di utilizzo.

Eliminata la parte che richiama la ricerca con ms “piccoli” nell’evento della selezione del tasto Enter,

if ((long.Parse(this.searchMs.Text) >= this.SliderMs.Minimum) && (long.Parse(this.searchMs.Text) <= long.Parse(maxMs.Content.ToString())))

{

//Lunghezza maggiore di 0 e valore inferiore al massimo

shortMsResearch(searchedValue - (long)this.SliderMs.Minimum, myDepoData);

}

Else

8.12:

Scritta la classe CncResult che contiene la lista con le grandezze misurate nell’esperimento e una lista con i rispettivi valori.

Tolte le seguenti funzioni:

private string extractPowerFeedback(string cncResult)

{

string powerFeedback = cncResult;

int index;

//The hardcoded way...

for (int i = 0; i < 15; i++)

{

index = powerFeedback.IndexOf("\t") + "\t".Length;

powerFeedback = powerFeedback.Substring(index);

}

if (!powerFeedback.Any())

{

powerFeedback = "No value";

}

return powerFeedback;

}

private string extractLaserOn(string cncResult)

{

string laserOn = cncResult;

int index;

//The hardcoded way...

for(int i=0; i<14; i++)

{

index = laserOn.IndexOf("\t") + "\t".Length;

laserOn = laserOn.Substring(index);

}

if (!laserOn.Any())

{

laserOn = "No value";

}

else

{

index = laserOn.IndexOf("\t");

laserOn = laserOn.Substring(0, index);

}

return laserOn;

}

Corretta la classe CncResult 🡪 funziona!

Sistemazione dell’interfaccia grafica: tolti ms significativi e checkBox per mostrarli. Massimizzazione della finestra.

Sistemazione di un bug: di volta in volta si toglieva la prima linea del file cn. Risolto copiando la lista in una nuova evitando modifiche sull’originale.

Notato un piccolo bug nella ricerca: non sembra caricare l’immagine trovata 🡪 controloare

9.12:

Ora gli esperimenti in lista sono selezionabili con il mouse e navigabili con le frecce.

ScrollViewer e StackPanel sostituiti con una ListBox 🡪

<ListBox x:Name="ExperimentViewer">

</ListBox>

In MainWindow.xaml.cs 🡪

ExperimentViewer.Children.Add(listItem); è diventato ExperimentViewer.Items.Add(listItem);

La visualizzazione del file txt dell’esperimento si raggiunge tramite il container webBrowser, ma prende tutta la colonna degli esperimenti.

Purtroppo non riesco ancora a caricare il txt nel documentViewer, non comprendo in che modo funzioni la proprietà .document.

Aggiunta la toolbar e il pulsante di refresh, ancora da implementare..

<ToolBar Grid.Row="0" Grid.ColumnSpan="4">

<Button Width="80">(Reset Layout)</Button>

<Button Width="80">(Close Tabs)</Button>

<Button Width="80">(Refresh)</Button>

</ToolBar>

10.12:

Abbandonato per il momento la via del DocumentViewer a favore del control TextBox 🡪 leggendo le righe dal file si riescono a visualizzare nel texBox. ExpComment.Text = File.ReadAllText(fileName);

ANCORA DA GESTIRE EVENTUALI ECCEZIONI SU FILE NON TROVATO!!

Il documentviewer richiede un fixeddocument, il quale collabora con PageContent e FixedPage, tutte e tre non sembrano avere metodi che accettano stringhe o file da mostrare… Tuttavia sembra l’opzione più carina viste alcune sue chicche, quali la ricerca del testo ecc…

1. 10.12-17.12  
   Incontro:  
     
   1. Aggiungere i ms minimi accanto agli attuali sotto l’immagine  
   2. Aggiungere i ms della temperatura e separare i dati del pirometro da quelli del CN  
   3. BUG Michele ha aperto due esperimenti, ognuno con una deposizione aperta, dopo aver aperto il secondo esperimento è tornato sul primo  
   4. Aprire più deposizioni uguali 🡪 confronto! Cambiare assolutamente il sistema di chiave valore!!!!

Inoltre ci sono le pendenze degli ultimi meeting:  
  
Prima del 6.12:  
1.      ToDo Gestire sia depo folder con vecchia struttura che nuova  
2.      ToDo Chiudere experiment tab  
3.      ToDo Ricordare il path dell'ultimo experiment aperto, e ultima depo aperta  
=> Aggiungere un menu all' experiment, e uno alla depo, e salvare nel file .ini l'informazione  
=> Guardare come si fa in visual studio: non c'è da inventare nulla!

15.12:

OK Aggiungere i ms minimi accanto agli attuali sotto l’immagine

* + Aggiunta label nello user control
  + Aggiunta del codice per impostare la label

private Label extMinMs;

this.extMinMs = depoItemBody.ExtendMinMs;

this.extMinMs.Content = minString;

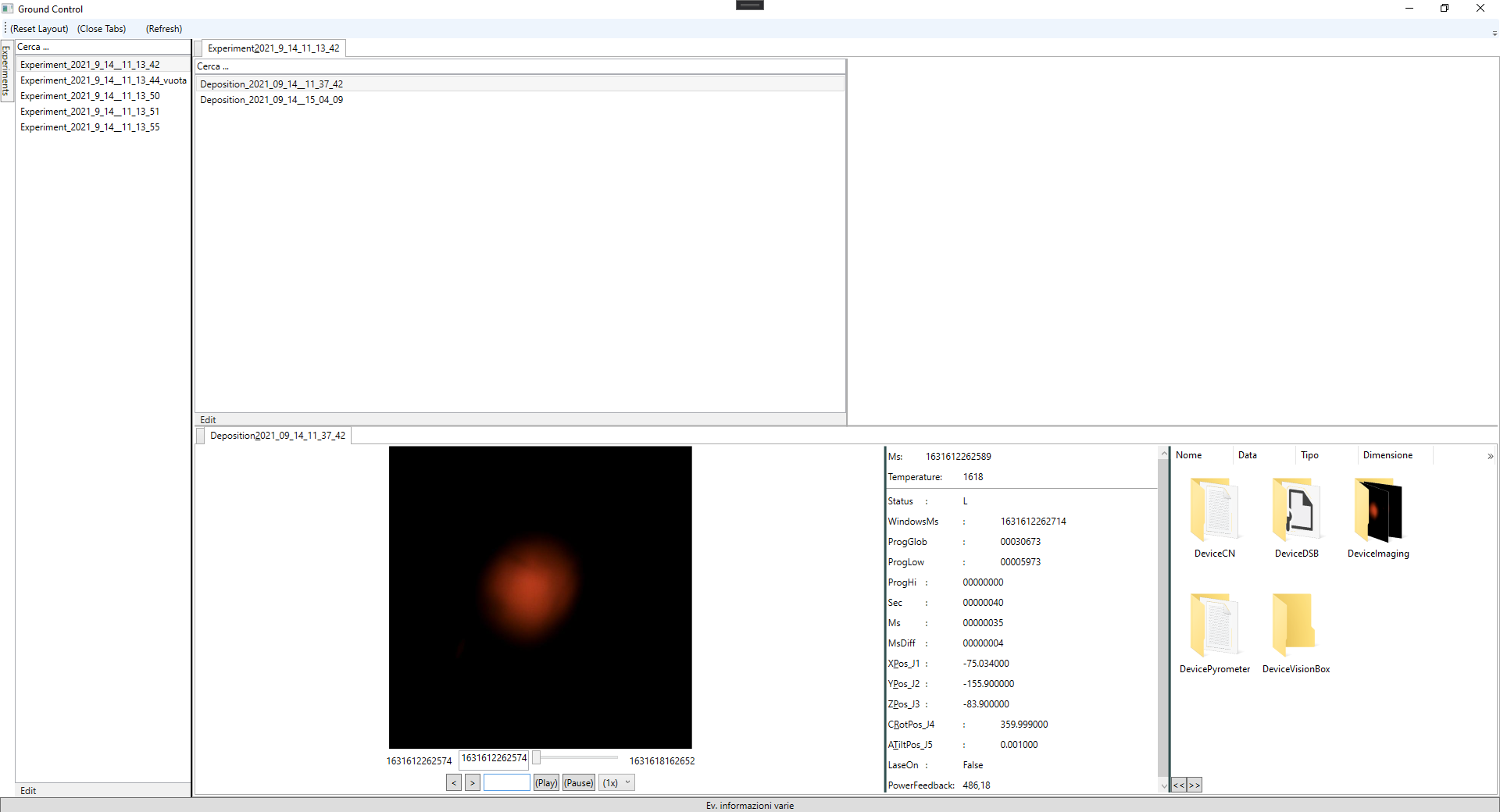
Risolto il bug della ricerca immagini: non settava l’immagine. Questo poiché il seguente comando

* myDepoData.setActualImage(myDepoData.getImages().IndexOf(filename));

era posto in fondo alla funzione setImage(string filename) anziché fra i primi comandi della stessa.

Aggiunta una classe che wrappa i ms di riferimento per la temperatura e il valore della temperatura.

OK Aggiungere i ms della temperatura e separare i dati del pirometro da quelli del CN



16.12:

Si cerca di aggiungere la funzionalità della ricerca al termine del drag del cursore dello slider.

A livello di interfaccia nel tag dello Slider è stato aggiunto Thumb.DragCompleted="ExtendMsSlider\_DragCompleted"

https://stackoverflow.com/questions/723502/wpf-slider-with-an-event-that-triggers-after-a-user-drags

Modifica della firma della classe DepoTabControlController, poiché passandolo come parametro al costruttore di DepoItemBody si otteneva un errore del compilatore CS0051.

public class DepoTabControlController

In cascata anche la seguente classe public class MyExpTabItemModel.

Modifiche di accessibilità

public String getDepoName()

Metodo dell’evento relativo alla fine del drag del cursore dello slider

//Evento collegato al cursore dello slider, che richiama una funzione della classe DepoTabControlController

private void ExtendMsSlider\_DragCompleted(object sender, System.Windows.Controls.Primitives.DragCompletedEventArgs e)

{

//Variabile locale che ottiene il valore dallo slider --> cast a long poiché i valori predefiniti dello slider sono double

long searchedValue = (long)ExtendMsSlider.Value;

//Variabile locale che ottiene il nome della deposizione, utile per richiamare l'istanza MyDepoData da passare alla funzione di ricerca

String depositionName = this.depoTabControlController.getDepoName();

//Variabile locale che ottiene l'istanza myDepoData di riferimento

MyDepoData myDepoData = this.depoTabControlController.getDepoDatas()[depositionName];

//Ricerca del valore tramite l'istanza depoTabControlController

this.depoTabControlController.msResearch(searchedValue, myDepoData);

}

Momentaneamente la funzionalità è "limitata" al movimento del cursore, ma non al cambiamento di valore dovuto al click sullo slider. Il valueChange potrebbe creare problemi durante i drag…

17.12:

Per chiudere le tab provo a seguire il seguente tutorial/esempio: <https://www.codeproject.com/Articles/84213/How-to-add-a-Close-button-to-a-WPF-TabItem>

Sembra sia necessario creare una classe, una TabItem, su misura.

Gli steps:

* Create a new UserControl and call it CloseableHeader. OK
* Add a Label to this control and call it label\_TabTitle. OK
* Add a Button to this control and call it button\_close. OK
* Set the style of the button to ToolBar.ButtonStyleKey. OK
* Set  the Text (content) of the button to **X**.

Manca la seguente parte:

ClosableTab theTabItem = new ClosableTab();

theTabItem.Title = "Small title";

tabControl1.Items.Add(theTabItem);

theTabItem.Focus();

La quale potrebbe dare diversi errori quando si cerca di recuperare l’header della tabItem, poiché conterrà, oltre al testo utile, anche il bottone.

Si sono presentati degli errori in effetti e sono stati eseguiti dei cast, ma sembra funzionare Sìiiiiii!!!!!!

Aggiunto il .Title get al codice della CloseableTab:

public string Title

{

set

{

((CloseableHeader)this.Header).label\_TabTitle.Content = value;

}

get

{

return (string)((CloseableHeader)this.Header).label\_TabTitle.Content;

}

}

//Controllo dell'istanza, se si tratta di un CloseableTab allora si esegue il cast

if (mainTabControl.SelectedItem is CloseableTab)

{

CloseableTab tabItem = (CloseableTab)mainTabControl.SelectedItem;

//Mandatory check to avoid tabItem=null happened on drag&drop the tabItem

//Controllo obbligatorio per evitare che il drag&drop della tab produca un tabItem con riferimento null (ACCADEVA!)

if (tabItem != null)

{

//Mandatory to open the deposition in the correct tabControl

//Necessario affinché la deposizione sia aperta nel corretto TabControl

String header = (string)tabItem.Title;

if (header != null)

{

//depoTabControlController.setDepoPath((string)tabItem.Header);

//depoTabControlController.setActualTabControl((string)tabItem.Header);

depoTabControlController.setDepoPath(header);

depoTabControlController.setActualTabControl(header);

}

}

}

21.12:

Modificata la funzione dell’evento della SelectedItem sul TabControl della deposizione in egual modo (vedi sopra).

Sistemazione dei vari errori riscontrati durante il test di funzionamento dei TabItem.

Sostituzione di TabItem con la classe CloseableTab anche nella tabControl delle deposizioni.

26.12:

Ricerca sui delegate, per capire se possono essere utili. Continuo a non esserne certo.

28.12:

Creata la versione lite degli esperimenti, dopodiché si inizia a lavorare sul commento dell’esperimento e sull’immagine (provino).

Risolto il seguente bug 🡪 quando si cambiava la tab dell’esperimento e si cliccava sulla prossima immagine nella deposizione già selezionata l’app crashava, poiché non si trovavano i dati corretti (non avveniva alcuna selezione a livello di deposizione). Perciò è stato aggiunto lo stesso settaggio dell’evento del cambio del TabItem.

//Richiama il TabItem selezionato

if (actualTabControl.SelectedItem is CloseableTab)

{

CloseableTab tabItem = (CloseableTab)actualTabControl.SelectedItem;

String header = (string)tabItem.Title;

//Mandatory check to avoid tabItem=null happened on drag&drop the tabItem

if (tabItem != null && header != null)

{

//Si richiama la depoItemBody di riferimento dalla struttura dati per l'esperimento selezionato

DepoItemBody depoItemBody = this.depoStructures[header];

//Verifica che essa non sia null

if (depoItemBody != null)

{

//Si assegnano i relativi controlli (bottoni, slider, ecc...) e il percorso dell'esperimento nel file system

assignIControl(depoItemBody);

setDataPath(header);

}

}

}

29.12:

Aggiornata la versione di Visual Studio alla 16.11.8

Inserito un FlowDocumentReader per mostrare il contenuto del file del commento 🡪 si vede!

Il codice usato preso dal sito 🡪 <https://www.c-sharpcorner.com/uploadfile/mahesh/loading-a-text-file-in-a-wpf-flowdocumentreader/>

Paragraph paragraph = new Paragraph();

paragraph.Inlines.Add(System.IO.File.ReadAllText(fileName));

FlowDocument document = new FlowDocument(paragraph);

ExpDetails.Height = this.ActualHeight / 2;

Provino.Height = ExpDetails.ActualHeight / 2;

ExpComment.Height = ExpDetails.ActualHeight / 2;

ExpComment.Document = document;

A livello di interfaccia grafica è stato inserito uno stackpanel che contiene un control image e il FlowDocumentReader. È l’intero stackpanel ad apparire e scomparire al click dal menu.

Aggiunto un bordo all’immagine: <https://stackoverflow.com/questions/3092992/how-do-i-put-a-border-around-an-image-in-wpf>

30.12:

Drag and Drop dell’immagine 🡪 https://stackoverflow.com/questions/5662509/drag-and-drop-files-into-wpf

Al momento il drag and drop funziona se esso avviene sulla lista degli esperimenti, sul commento o sulla zona dell’immagine non funziona 🡪 si dovrà capire il perché.

Inserita una finestra di conferma per il cambiamento dell’immagine 🡪 <https://stackoverflow.com/questions/3036829/how-do-i-create-a-message-box-with-yes-no-choices-and-a-dialogresult>

Apertura della directory degli esperimenti o dell’esperimento selezionato nel file explorer di Windows.

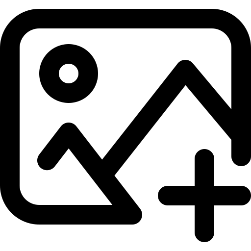
Impiegata una funzione generica per l’avvio dell’explorer

<https://www.codeproject.com/Questions/852563/How-to-open-file-explorer-at-given-location-in-csh>

Scaricate immagini di placeholder per i provini

<div>Icons made by <a href="https://www.freepik.com" title="Freepik">Freepik</a> from <a href="https://www.flaticon.com/" title="Flaticon">www.flaticon.com</a></div>

<div>Icons made by <a href="https://www.flaticon.com/authors/good-ware" title="Good Ware">Good Ware</a> from <a href="https://www.flaticon.com/" title="Flaticon">www.flaticon.com</a></div>



Creata una cartella nel progetto per le immagini sopra mostrate.

Inizio dell’implementazione del comando di “refresh”:

<https://stackoverflow.com/questions/4773632/how-do-i-restart-a-wpf-application>

System.Diagnostics.Process.Start(Application.ResourceAssembly.Location);

Application.Current.Shutdown();

Implementazione comando di chiusura dal menu “File” 🡪 <https://stackoverflow.com/questions/2820357/how-do-i-exit-a-wpf-application-programmatically>

System.Windows.Application.Current.Shutdown();

31.12:

Modificato l’evento della selezione dell’item in lista degli esperimenti. Con l’evento che si aveva sinora, //listItem.Selected += ListItem\_Selected;, si aveva il problema al cambio dell’item selezionato, poiché catturava ancora i dati del vecchio item. L’evento è stato passato al contenitore ListBox con un SelectionChanged 🡪 ora funziona!

Apertura del file del commento tramite il programma di default affinché possa essere modificato.

<https://stackoverflow.com/questions/11365984/c-sharp-open-file-with-default-application-and-parameters>

System.Diagnostics.Process.Start(commentPath);

Inserito il controllo per avere unicamente cartelle con gli esperimenti nella lista di sinistra e unicamente deposizioni nella cartella dell’esperimento che si sta aprendo.

string depoName = depoDirectories[i].Remove(0, depoPath.Length + 1);

if (depoName.Contains("Deposition"))

{

listItem.Content = depoName;

listItem.MouseDoubleClick += depoTabControlController.openDepsData;

depoList.Items.Add(listItem);

}

string folderName = expDirectories[i].Remove(0, expPath.Length + 1);

//Si verifica che il nome contenga la parola "Experiment"

if (folderName.Contains("Experiment"))

{

//Si assegna il nome al content del ListViewItem

listItem.Content = folderName;

//Alla ListViewItem si aggiunge l'evento openExpDeps (l'evento che permette di aprire la tab dell'esperimento)

listItem.MouseDoubleClick += openExpDeps;

//Si aggiunge l'elemento della lista appena creato alla viewer degli esperimenti

ExperimentViewer.Items.Add(listItem);

}

8.01:

Inseriti alcuni controlli:

* Controlla se ci sono esperimenti nella lista, altrimenti inserisce una label di avviso come elemento  
  //Controllo se la lista degli esperimenti contiene elementi --> se no, inserisce la label di comunicazione

private void checkEmptyExpList(ListBox experimentViewer)

{

bool hasElement;

//Controllo se la cartella indicata contiene cartelle relative agli esperimenti

if (experimentViewer.Items.Count > 0)

{

hasElement = true;

}

else

{

hasElement = false;

}

if (!hasElement)

{

//Se la lista non contiene elementi, ossia nessuna cartella di esperimenti, sarà mostrata la scritta di avviso

Label label = new Label();

label.Content = noExpDetected;

experimentViewer.Items.Add(label);

}

}

* Controlla se l’elemento è un listviewitem, se sì esegue le operazioni relative al comando  
   //Verifica se l'item selezionato della lista è un listviewitem --> se sì torna true

private bool checkItemType()

{

if(ExperimentViewer.SelectedItem is ListViewItem)

{

return true;

}

else

{

return false;}}

Creazione e implementazione della classe di controllo del file di configurazione. Informazioni sul codice reperite dai seguenti siti:

<https://www.codeproject.com/Articles/14744/Read-Write-App-Config-File-with-NET-2-0>

<https://stackoverflow.com/questions/4758598/write-values-in-app-config-file>

<https://stackoverflow.com/questions/10766654/appsettings-get-value-from-config-file>

Test scrittura e lettura dal file di configurazione 🡪 ok!

Salvataggio dell’immagine di provino all’interno della cartella dell’esperimento durante il drop, con il seguente formato e nome: “Esperimento31651\_Provino.jpg”.

Fonte per il salvataggio dell’immagine: <https://docs.microsoft.com/en-us/dotnet/api/system.drawing.image.save?view=dotnet-plat-ext-6.0>

Convertitore bitmapimage a bitmap: <https://stackoverflow.com/questions/6484357/converting-bitmapimage-to-bitmap-and-vice-versa>

Se non c’è l’immagine nella cartella dell’esperimento, si carica l’immagine di default contenuta nella cartella immagini. new Uri(@"/Immagini/GC\_addImage.png", UriKind.RelativeOrAbsolute)

Caricare solo immagini jpeg!!

10.01:

Tentativo di utilizzo dei task per mostrare le immagini in automatico e sfruttarne l’interruzione tramite un token.

Prendere il valore dal combobox per la “velocità”: <https://stackoverflow.com/questions/4351603/get-selected-value-from-combo-box-in-c-sharp-wpf>

Il task causa un system.invalidoperationexception sulla bitmapImage.

11.01:

Accade lo stesso con un thread, poiché l’oggetto in questione è sotto il controllo di un altro thread (interfaccia grafica).

La serata è stata investita in ricerche infruttuose…

12.01:

La classe DSB\_Controller è stata tolta, le sue responsabilità sono state ridistribuite correttamente: non era necessario avere una classe in più per i compiti che svolgeva.

13.01:

<https://stackoverflow.com/questions/42147076/c-sharp-wpf-change-imagesource-image-on-thread>

Spostato il controllo legato alle immagini al cs corretto, quello legato allo xaml dell’interfaccia dei controlli della deposizione.

LA SOLUZIONE ALLE IMMAGINI AUTOMATICHEEEEE, niente thread..

<https://stackoverflow.com/questions/33298348/update-gui-in-a-loop>

private async void PlayImage\_Click(object sender, RoutedEventArgs e)

{

MessageBox.Show("Bottone play", "Bottone schiacciato");

isAutomatic = true;

while(isAutomatic)

{

int ratio = getSpeed();

nextImage();

if (areImageEnd())

{

isAutomatic = false;

}

await Task.Delay(timeMs/ratio);

}

}

13-14.01:

Aggiunto l’indice delle copie per quanto concerne l’apertura di più esperimenti/deposizioni.

Modalità 🡪 Experiment12, Experiment12(1), …

Esperimenti

public bool Contains(ListViewItem itemID)

{

int copyindex = 0;

string temp = (string)itemID.Content;

try

{

while (items.Contains(temp))

{

copyindex++;

temp = (string)itemID.Content + "(" + copyindex.ToString() + ")";

}

itemID.Content = temp;

return false;

}

catch(Exception exception)

{

return true;

}

}

Deposizioni

private Boolean allowAdding(ListViewItem listViewItem)

{

int copyindex = 0;

string temp = (string)listViewItem.Content;

try

{

while (this.depoStructures.ContainsKey(temp))

{

copyindex++;

temp = (string)listViewItem.Content + "(" + copyindex.ToString() + ")";

}

listViewItem.Content = temp;

return true;

}

catch (Exception exception)

{

return false;

}

}

Inoltre serve un metodo per estrarre il nome dell’esperimento o della deposizione, a seconda di cosa si tratta. Il metodo restituisce la stringa escludendo l’eventuale segno di copia: es. “(3)”. Così facendo è possibile istanziare ed accedere ai dati corretti.

Experiment12(2) 🡪 Experiment12

private string extractName(string header)

{

if (header.Contains("("))

{

int endIndex = header.IndexOf("(");

string name = header.Substring(0, endIndex);

return name;

}

else

{

return header;

}

}

16.01:

Il file di configurazione è il seguente " \bin\Debug\DataSetBuilder.exe.Config".

Sistemata la parte legata al percorso temporaneo. Quando l’utente nega alla domanda, nel file si inserisce una chiave legata al percorso temporaneo. Al riavvio il programma controlla se è presente un percorso temporaneo e quindi avvia quello anziché il percorso predefinito.

//Ripresa dei dati sul percorso dal file di configurazione

private void PathFromConfig()

{

//Verifica se il file di configurazione contiene un percorso segnato come temporaneo

if (configurationController.containsTemppath())

{

//Imposta il percorso dell'applicativo al percorso temporaneo di configurazione

this.expPath = configurationController.getTempPath();

}

//Verifica se il file di configurazione contiene un percorso segnato come predefinito

else if (configurationController.containspath())

{

//Imposta il percorso dell'applicativo al percorso predefinito di configurazione

this.expPath = configurationController.getConfigPath();

}

else

{

//Richiama la funzione per impostare un nuovo percorso se non sono trovati ne quello predefinito ne quello temporaneo

changeExpPath();

}

}

Aggiunta la domanda sulla chiusura dell’applicazione dalla X, inoltre all’interno delle chiusure è stato inserito un comando per togliere il percorso temporaneo dal file di configurazione, di modo che non lo si ritrova durante la prossima apertura dell’applicativo 😊

//Si chiude l'applicazione e si toglie il riferimento al percorso temporaneo

if (configurationController.containsTemppath())

{

string key = "temppath";

configurationController.remove(key);

}

L’evento di chiusura è stato disattivato quando l’applicativo viene riavviato o chiuso in modo diverso rispetto alla X, poiché altrimenti l’evento si verificherebbe ugualmente:

this.Closing -= Window\_Closing;

Separato il comando da menu di aggiornamento degli esperimenti e del riavvio dell’app. Nell’aggiornamento è stato semplicemente richiamato l’inizializzazione della lista degli esperimenti.

Durante una prova è stato generato un ArgumentException, poiché a quanto pare nel file di configurazione è presente la chiave “temppath” ma non ha alcun valore… L’eccezione è stata catturata con un catch e gestita come la DirectoryNotFoundException 🡪 si chiama il cambio di percorso.

Aggiunto la gestione in caso l’utente prema “Annulla” dalla finestra di selezione della cartella, <https://stackoverflow.com/questions/9567061/folderbrowserdialog-cancel-event-c-sharp>

Scaricata immagine nel caso non si trovino immagini durante lo scorrimento delle deposizioni, poiché altrimenti va in crash <https://unsplash.com/s/photos/sorry>

18.01:

Aggiunta della ricerca dinamica nel campo di testo della lista degli esperimenti, e della lista delle deposizioni, chiamando un evento ogniqualvolta il testo viene modificato. Alla chiamata di tale metodo si richiama l’inizializzazione della listbox applicando un ulteriore filtro al nome delle cartelle riportate nella lista. Conseguentemente è stato applicato un parametro al metodo Init(int number) al fine di poter applicare un costrutto switch all’interno di Init e applicare le corrette operazioni.

Per la parte delle deposizioni la chiamata di inizializzazione è stata spostata in expitem.cs da myExpTabItemFactory; la quale probabilmente sarà cancellata al prossimo refactoring.

Esempio con la parte dell’esperimento, ma è analogo per la deposizione:

//Evento che si verifica quando si modifica il testo nella searchbox degli esperimenti --> in cima alla lista

private void ExpSearchBox\_TextChanged(object sender, TextChangedEventArgs e)

{

//Aggiornamento della lista

Init(1);

}

//Switch per gestire l'inizializzazione iniziale e l'aggiornamento dinamica della listbox degli esperimenti digitando nella casella di ricerca

switch (number)

{

//Inizializzazione

case 0:

//Si verifica che il nome contenga la parola "Experiment"

if (folderName.Contains("Experiment"))

{

//Si assegna il nome al content del ListViewItem

listItem.Content = folderName;

//Alla ListViewItem si aggiunge l'evento openExpDeps (l'evento che permette di aprire la tab dell'esperimento)

listItem.MouseDoubleClick += openExpDeps;

//Si aggiunge l'elemento della lista appena creato alla viewer degli esperimenti

ExperimentViewer.Items.Add(listItem);

}

break;

//Aggiornamento tramite searchbox

case 1:

string textsearched = ExpSearchBox.Text;

//Si verifica che il nome contenga la parola "Experiment"

if (folderName.Contains("Experiment") && folderName.Contains(textsearched))

{

//Si assegna il nome al content del ListViewItem

listItem.Content = folderName;

//Alla ListViewItem si aggiunge l'evento openExpDeps (l'evento che permette di aprire la tab dell'esperimento)

listItem.MouseDoubleClick += openExpDeps;

//Si aggiunge l'elemento della lista appena creato alla viewer degli esperimenti

ExperimentViewer.Items.Add(listItem);

}

break;

}

Aggiunto il comando di aggiornamento della lista delle deposizioni in una voce del menu delle opzioni della deposizione. Anche la stessa voce degli esperimenti è stata spostata dal menu File nella toolbar al menu delle opzioni degli esperimenti.

Aggiunti anche i tag xml per l’eventuale immagine e l’eventuale commento legati alla deposizione selezionata.