|  |  |
| --- | --- |
| GUI per applicativo di Data Collection – Ground Control | |
|  | |
| Studente/i | Relatore |
| Mariano Fasano | Michele Banfi |
| Correlatore |
| - |
| Committente |
| - |
| Corso di laurea | Modulo |
| Ingegneria Informatica | Progetto di semestre |
| Anno |  |
| 2021-2022 |  |

|  |  |
| --- | --- |
| Data |  |
| - |  |

Indice generale

[1 Introduzione 10](#_Toc94638453)

[2 Analisi 12](#_Toc94638454)

[2.1 Casi d’uso 12](#_Toc94638455)

[2.2 Libreria grafica 15](#_Toc94638456)

[2.3 Analisi di C# e WPF 19](#_Toc94638457)

[2.4 Dati 20](#_Toc94638458)

[3 Tecnologie Utilizzate 21](#_Toc94638459)

[3.1 Visio e Pencil 21](#_Toc94638460)

[3.2 Visual Studio 21](#_Toc94638461)

[3.3 C# 21](#_Toc94638462)

[3.4 Github e Github Desktop 21](#_Toc94638463)

[4 Sviluppo 22](#_Toc94638464)

[4.1 Mockup 22](#_Toc94638465)

[4.2 User-control 24](#_Toc94638466)

[4.3 Classi e metodi rilevanti 25](#_Toc94638467)

[5 Analisi dei risultati 25](#_Toc94638468)

[5.1 Risultato 25](#_Toc94638469)

[5.2 Manuale d’uso 27](#_Toc94638470)

[5.3 Problemi risolti 35](#_Toc94638471)

[5.3.1 File Explorer 35](#_Toc94638472)

[5.3.2 Binary Search 36](#_Toc94638473)

[5.3.3 Errore iniziale 38](#_Toc94638474)

[5.3.4 Automazione delle immagini mostrate 38](#_Toc94638475)

[5.3.5 Chiusura delle tab 39](#_Toc94638476)

[5.3.6 Refactoring limitato sullo user-control 42](#_Toc94638477)

[5.3.7 Impossibilità di separare fisicamente file xaml e relativo cs 43](#_Toc94638478)

[5.4 Sviluppi futuri 43](#_Toc94638479)

[5.4.1 Parte online e collegamento al database 43](#_Toc94638480)

[5.4.2 Gestione del click sullo slider 43](#_Toc94638481)

[5.4.3 I Dictionary 44](#_Toc94638482)

[5.4.4 Spostamento tab 44](#_Toc94638483)

[5.4.5 File di configurazione per i device 44](#_Toc94638484)

[5.4.6 Refactoring 44](#_Toc94638485)

[6 Conclusioni 46](#_Toc94638486)

[7 Fonti 49](#_Toc94638487)

[7.1 Bibliografia 49](#_Toc94638488)

[7.2 Sitografia 49](#_Toc94638489)

[7.2.1 Pagine 49](#_Toc94638490)

[7.2.2 Immagini 49](#_Toc94638491)

Indice delle figure

Figura 1: model-view controller 9

Figura 2: casi d'uso, legenda 11

Figura 3: caso d'uso, manage experiment 11

Figura 4: caso d'uso, manage deposition 12

Figura 5: caso d'uso, select component setup 12

Figura 6: caso d'uso, manage device setup 13

Figura 7: caso d'uso, export file from DB 13

Figura 8: caso d'uso, view acquired data 14

Figura 9: caso d'uso, start data acquisition 14

Figura 10: diagramma .NET 6 18

Figura 11: Mockup, presentazione finale del software 21

Figura 12: Ispirazione mockup, Visual Studio 22

Figura 13: Mockup, user-control relativo alla parte online 23

Figura 14: Mockup, parte online, creazione di un esperimento 23

Figura 15: Ground Control, all’apertura 24

Figura 16: Ground Control, at work 25

Figura 17: Manuale d'uso, GC aperto 26

Figura 18: Manuale d'uso, elenco esperimenti chiuso 26

Figura 19: Manuale d'uso, opzioni esperimento 27

Figura 20: Manuale d'uso, dettagli esperimento 28

Figura 21: Manuale d'uso, Drag&Drop provino 28

Figura 22: Manuale d'uso, conferma della nuova immagine provino 29

Figura 23: Manuale d'uso, nuova immagine provino 29

Figura 24: Manuale d'uso, opzioni deposizioni 30

Figura 25: Manuale d'uso, barra di ricerca 31

Figura 26: Manuale d'uso, deposizione aperta 31

Figura 27: Manuale d'uso, controlli immagine 32

Figura 28: Manuale d'uso, dati e esplora risorse 32

Figura 29: Manuale d'uso, menu File 33

Figura 30: Manuale d'uso, conferma cambio percorso 33

Figura 31: Manuale d'uso, esperimenti recenti 34

Figura 32: Manuale d'uso, conferma chiusura applicativo 34

Figura 33: BinarySearch, CNC 36

Figura 34: CloseableTab, eventi 40

Figura 35: user control, albero del progetto 42

Indice delle tabelle

Tabella 1: comparazione WPF e WinForms 16

Riassunto / Abstract

Il prodotto è un applicativo grafico dedicato, Ground Control, che permette all’utilizzatore di visualizzare i dati raccolti degli esperimenti svolti sul processo di deposizione del titanio. Ogni esperimento contiene delle deposizioni, i cui dati sono composti da immagini e da misure raccolte dai vari strumenti in istanti di tempo nel corso dell’esperimento.

L’obiettivo del software è offrire all’utente un’alternativa più funzionale alla consultazione dei risultati sul file system, la quale avviene aprendo e confrontando le immagini con applicazioni specifiche e ricercando un preciso istante di tempo per ricavare la temperatura dal file di testo del pirometro, per esempio.

Con la soluzione sviluppata nel progetto i confronti, le ricerche, l’aggiunta e la modifica delle informazioni sull’esperimento (immagine di provino e commento), la compressione in file zip di esperimenti e deposizioni, e altro ancora, sono tutte operazioni integrate in essa.

L’applicativo va a coprire la parte offline, legata agli esperimenti e alle deposizioni già svolte e i cui dati sono già stati nel database e sul file system, che l’utente si vedeva costretto a svolgere in maniera poco efficiente nel file system.

In un momento successivo a questo lavoro l’applicativo andrà a coprire anche la parte online, legata alla creazione di nuovi esperimenti secondo specifiche configurazioni già esistenti o da specificare, alla visualizzazione dello status dei device impiegati, alla visualizzazione dei dati in tempo reale durante l’acquisizione e alla possibilità di decidere se attivare manualmente o automaticamente l’acquisizione dei dati.

In futuro il prodotto andrà quindi a integrare e possibilmente a migliorare l’attuale applicativo “Data Set Builder” che si occupa della parte online.

The product is a dedicated graphical application, Ground Control, which allows the user to visualise the data collected from experiments carried out on the titanium deposition process. Each experiment contains depositions, the data of which are composed of images and measurements collected by the various instruments at instants of time during the experiment.

The aim of the software is to offer the user a more functional alternative to consulting the results on the file system, which is done by opening and comparing the images with specific applications and searching for a precise instant of time to obtain the temperature from the text file of the pyrometer, for example.

With the solution developed in the project, comparisons, searches, adding and editing information about the experiment (specimen image and commentary), compressing experiments and depositions into zip files, and more, are all integrated into it.

The application covers the offline part, related to the experiments and depositions that have already been carried out and whose data were already in the database and on the file system, which the user was forced to perform in an inefficient way in the file system.

At a later stage, the application will also cover the online part, linked to the creation of new experiments according to specific configurations that already exist or are to be specified, the display of the status of the devices used, the display of data in real time during acquisition and the possibility of deciding whether to activate data acquisition manually or automatically.

In the future, the product will therefore complement and possibly improve the current "Data Set Builder" application that deals with the online part.

Progetto assegnato

Il progetto di ricerca “Ground Control” del fondo nazionale svizzero si occupa di analizzare i parametri del processo di deposizione del titanio (stampanti 3D) e di ottimizzarne il controllo. Per collezionare al meglio i dati di processo vengono utilizzate telecamere imaging e termiche, pirometri, e misure della macchina a controllo numerico “CN”. Nel corso del progetto, l’obiettivo della data collection è stato di scegliere le devices più adatte, integrarle nella macchina, interfacciarsi ad esse, e raccogliere i dati ed inserirli in un database relazionale MySQL (immagini, misure, protocolli, programmi, setup, ecc). A questo scopo è stato sviluppato l’applicativo “Data Set Builder”, aggiungendo ed integrando on demand le diverse funzionalità che si rendevano necessarie fra cui la graphical user interface. Questo progetto di semestre si pone l’obiettivo d’implementare un redesign dell’attuale GUI, la quale è stata sviluppata a spirale sulla base dei requisiti che nel corso del progetto si sono progressivamente delineati. Si richiede che la nuova GUI risulti completa, chiara e semplice da utilizzare, applicando le windows human interface guidelines. Il compito principale della GUI è di permettere all’utente di scegliere e caricare la configurazione desiderata per ogni device (gran parte del codice è già stato sviluppato per questa parte), e di visualizzarne l’output online (questa è la parte più consistente). Si sottolinea l’orientamento industriale e produttivo del presente progetto.

# Introduzione

Lo scopo di questo progetto, come citato nella voce “Progetto assegnato” in questo documento, è creare un applicativo denominato “Ground Control” che vada a gestire sia a un livello, da noi definito, offline, consultazione dei dati già acquisiti, sia a livello online, creazione di un nuovo esperimento (in seguito sarà spiegato meglio) con la relativa vista in tempo reale dei dati in acquisizione.

Uno degli obiettivi è rendere l’interfaccia intuitiva e usabile dall’operatore che eseguirà la raccolta dati dei vari esperimenti sul macchinario; l’operatore è da considerare un’entità del ramo tecnico, ma non un informatico o sviluppatore software.

Per realizzare l’interfaccia ci si orienta verso il pattern architetturale denominato Model-View Controller. Questo pattern, fondamentale anche nell’ambito web, implica l’uso di una parte di presentazione con la quale l’utente interagisce, l’interfaccia grafica con i relativi oggetti grafici. Interagendovi l’utente va a stimolare/scatenare una serie di eventi gestiti dalla parte logica/controllo del programma. Sulla base di questi stimoli essa va, se necessario, a consultare e a manipolare la parte di modello/dati contenente tutte le informazioni rilevanti dell’applicativo. Quando l’interrogazione dei dati è terminata, il controller li modifica se richiesto e restituisce la view/presentazione aggiornata all’utente tramite l’interfaccia grafica.

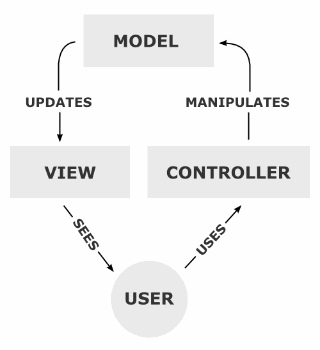


Figura 1: model-view controller

Nel caso specifico di questo progetto, nella parte offline per esempio, l’utente desidera aprire i risultati di una certa deposizione contenuta in un dato esperimento. Al doppio click del mouse, è la parte di controllo della view che si occupa di andare a interrogare i modelli che ha a disposizione, istanziandone altri relativi agli specifici dati richiesti in quel momento. Al termine delle operazioni l’utente vede una presentazione aggiornata secondo i comandi che ha fornito.

Si è pocanzi citata anche l’usabilità. Per il progetto bisogna naturalmente attenersi agli standard degli applicativi software in generale, chiamati anche Guidelines viste nei precedenti corsi e di recente nel corso di “Applicazioni Web 1”.

Sebbene non abbia mai impiegato questo software, un utente si aspetta alcuni comportamenti da parte sua: i nomi dei comandi devono rispecchiare la loro funzionalità (il comando “Aggiorna” dal menu degli esperimenti mi aggiorna la lista degli esperimenti, non l’intera interfaccia), e via dicendo.

Prima di iniziare il progetto sono stati completati i casi d’uso per avere un riassunto formale del comportamento che ci si aspetta dall’applicativo finale. Durante lo sviluppo del software ci si è resi conto, il sottoscritto e il relatore, prof. Banfi, che sarebbe stato necessario separare la parte offline di consultazione dei dati e la parte online per la creazione di nuovi esperimenti.

In primo luogo poiché sono emerse nuove specifiche nel corso dello sviluppo, altre invece sono andare a cadere oppure ci si è accorti che avrebbero avuto poco utilizzo se non poco senso.

In questa modalità vi è stato del concetto della metodologia Agile, poiché ogni settimana durante la riunione si potevano provare e testare nuove funzionalità, quindi anche un possibile committente/stackholder avrebbe visto l’applicativo crescere di incontro in incontro. Alcune funzionalità sono state suggerite dal team che lavora sul macchinario, una di queste è la possibilità di caricare un’immagine chiamata “Provino” che rendesse univoco o riconoscibile il dato esperimento. Da questo punto di vista unicamente la parte offline ha subito diversi cambiamenti di features. È stato preferito concentrarsi su di essa e implementare la parte online in un momento successivo a questo lavoro di progetto, poiché bisogna studiarne l’impiego al fine di progettarla in maniera adatta e collegarla a un database ancora in evoluzione.

Il metodo di lavoro usato è stato quello delle issues sfruttando le caratteristiche di github. Ogni settimana si teneva una riunione, della durata media di un paio di ore, nella quale si presentava il lavoro svolto, le difficoltà riscontrate, si valutavano nuove specifiche e la metodologia che si avrebbe voluto impiegare per implementarle, e via dicendo. I punti discussi erano trascritti sottoforma di elenco ordinato per priorità, se necessario, inviati via posta elettronica, inseriti nel “diario di bordo” del sottoscritto e tradotti in issues su github. Ovviamente quando emergevano dei punti critici o delle necessità ci si contattava via posta elettronica oppure via messaggio.

# Analisi

## Casi d’uso

I casi d’uso sono usati per schematizzare delle possibili situazioni di utilizzo dell’applicativo da parte dell’utente. Si evidenzia in modo chiaro quali azioni può compiere l’utente in determinate occasioni e quali, se caso, sono i compiti del sistema stesso.

Di seguito i casi di uso legati al progetto. Le parti interessate dal cambiamento avvenuto in corso d’opera riguardo la parte online[[1]](#footnote-1) sono state evidenziate.

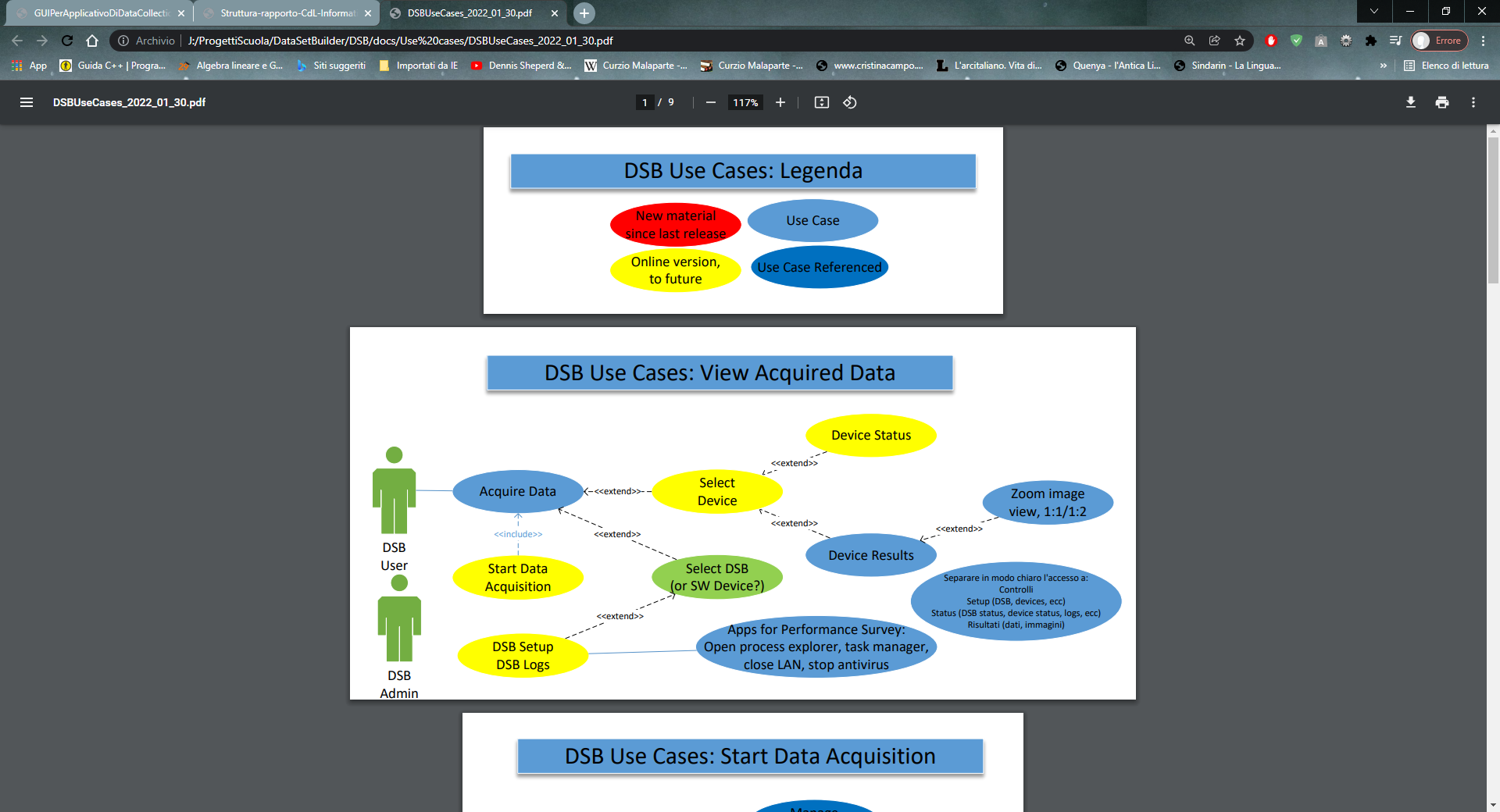


Figura 2: casi d'uso, legenda

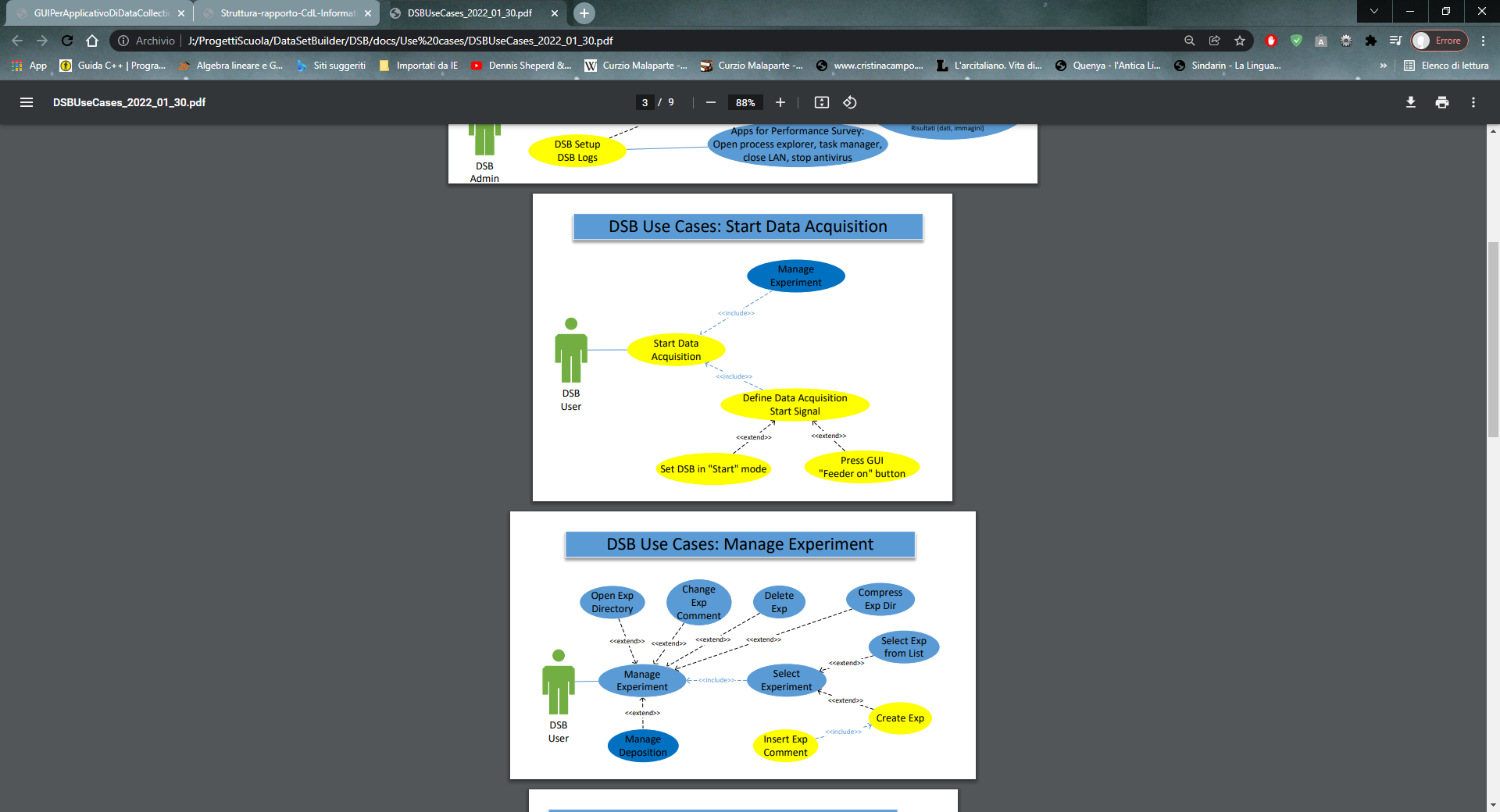


Figura 3: caso d'uso, manage experiment

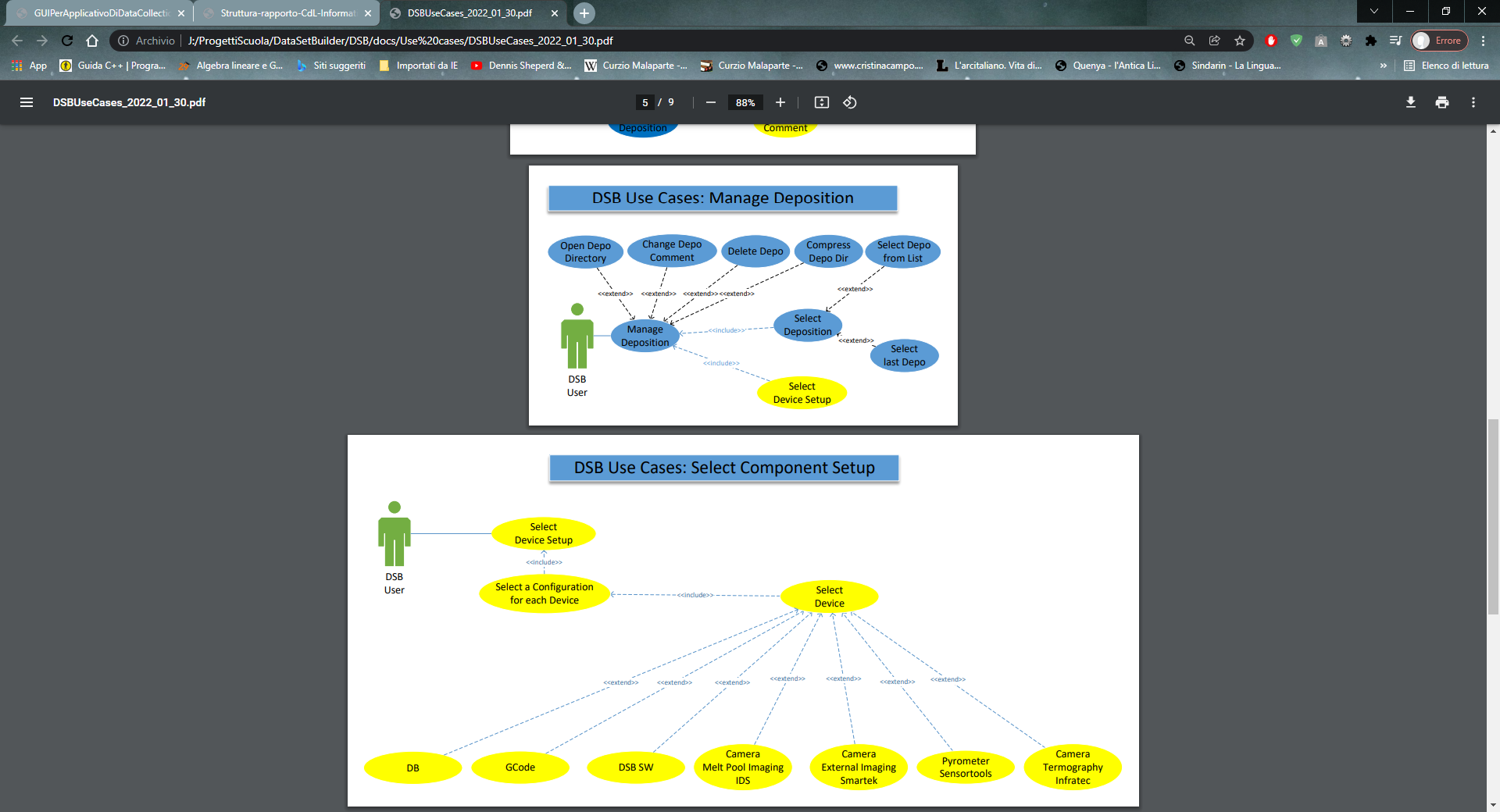


Figura 4: caso d'uso, manage deposition

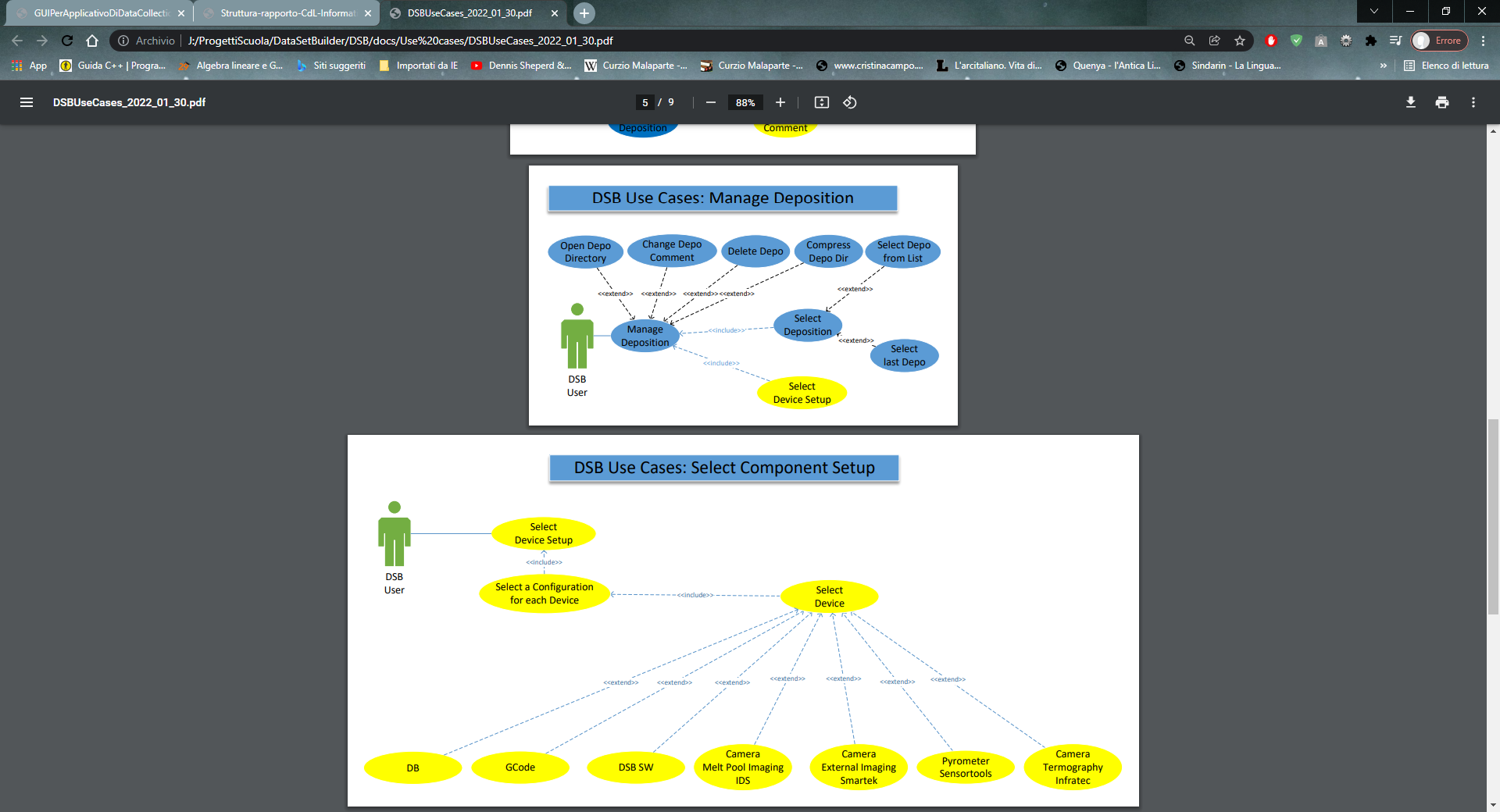


Figura 5: caso d'uso, select component setup

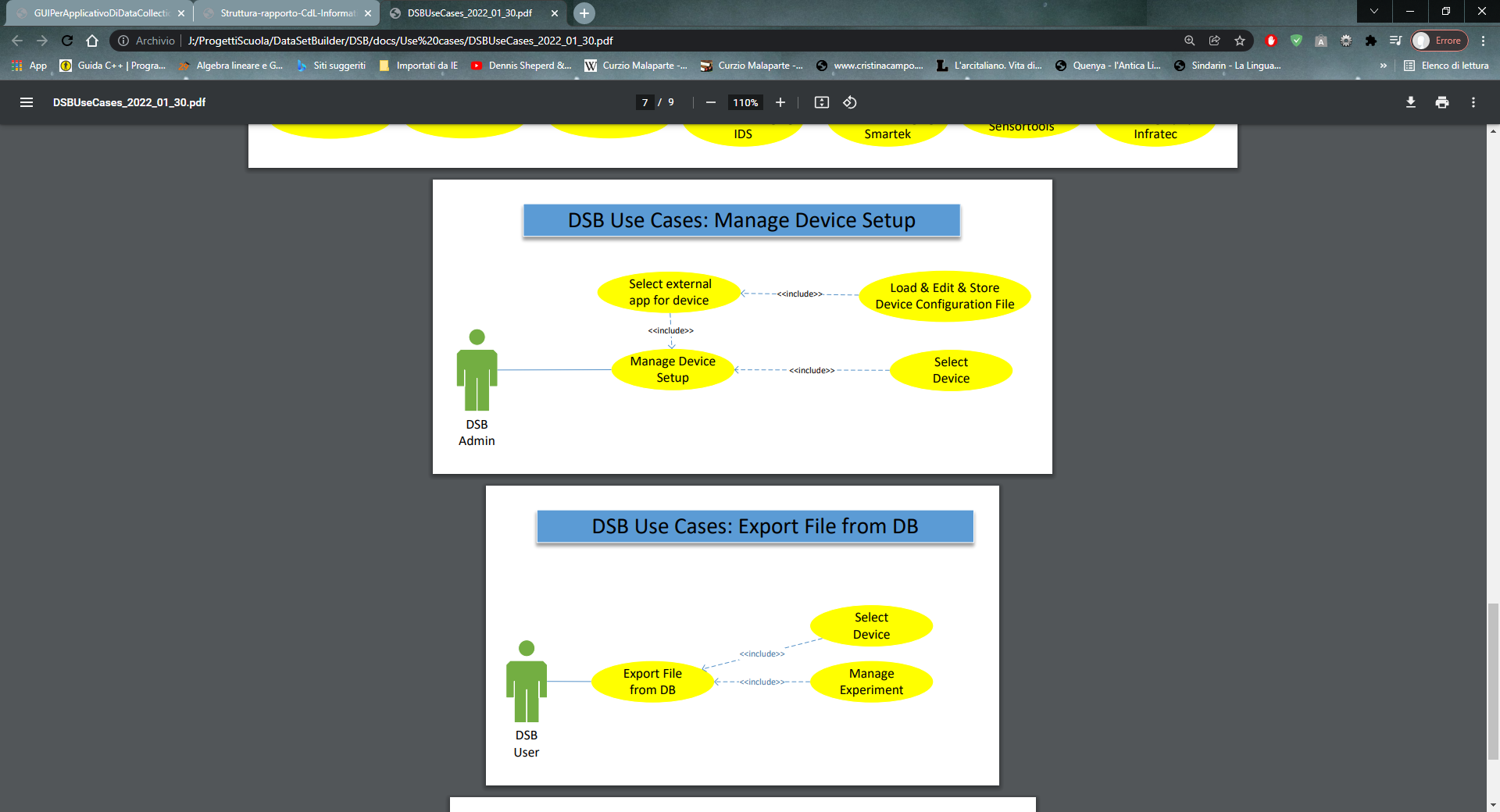


Figura 6: caso d'uso, manage device setup

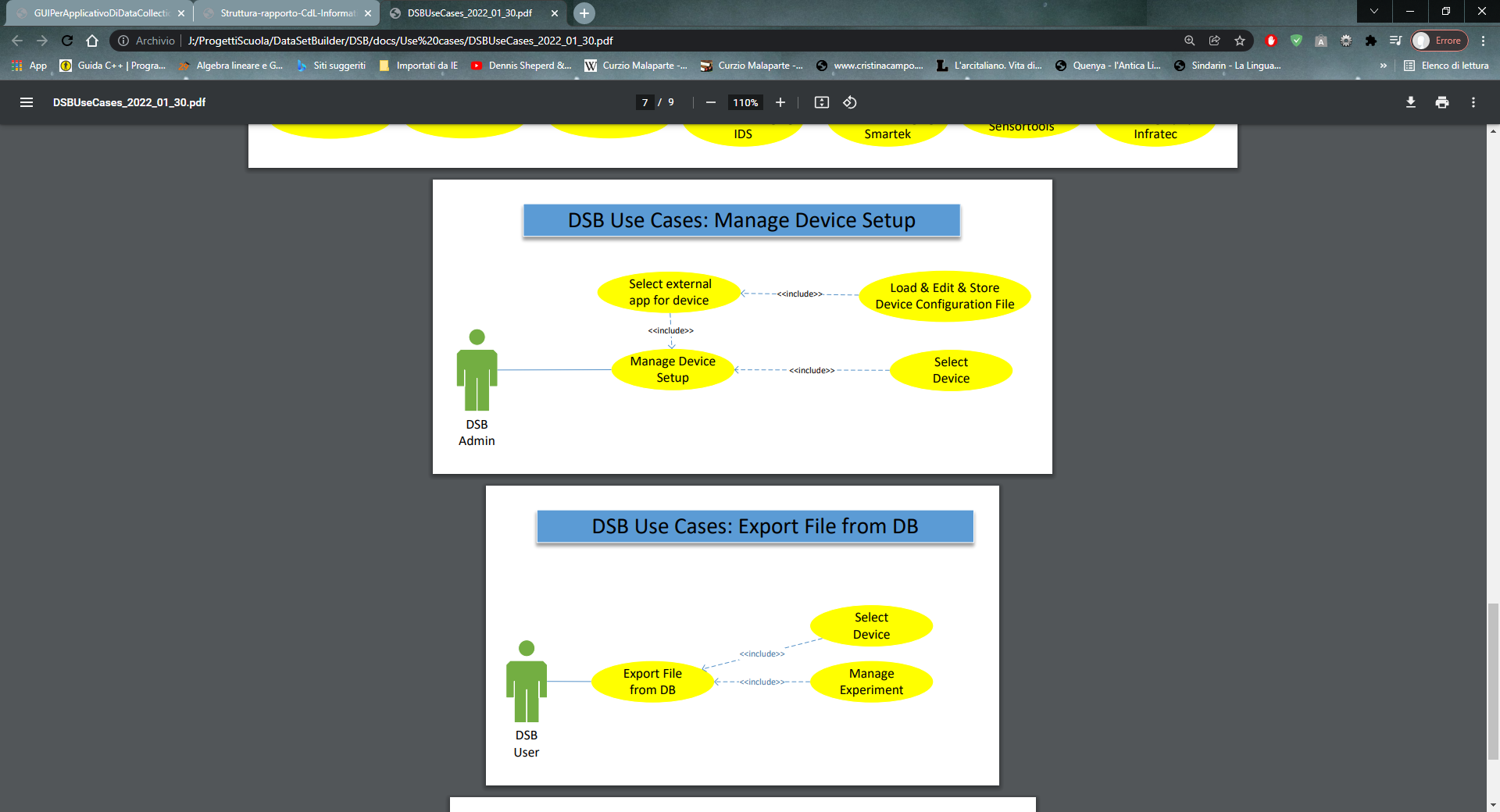


Figura 7: caso d'uso, export file from DB

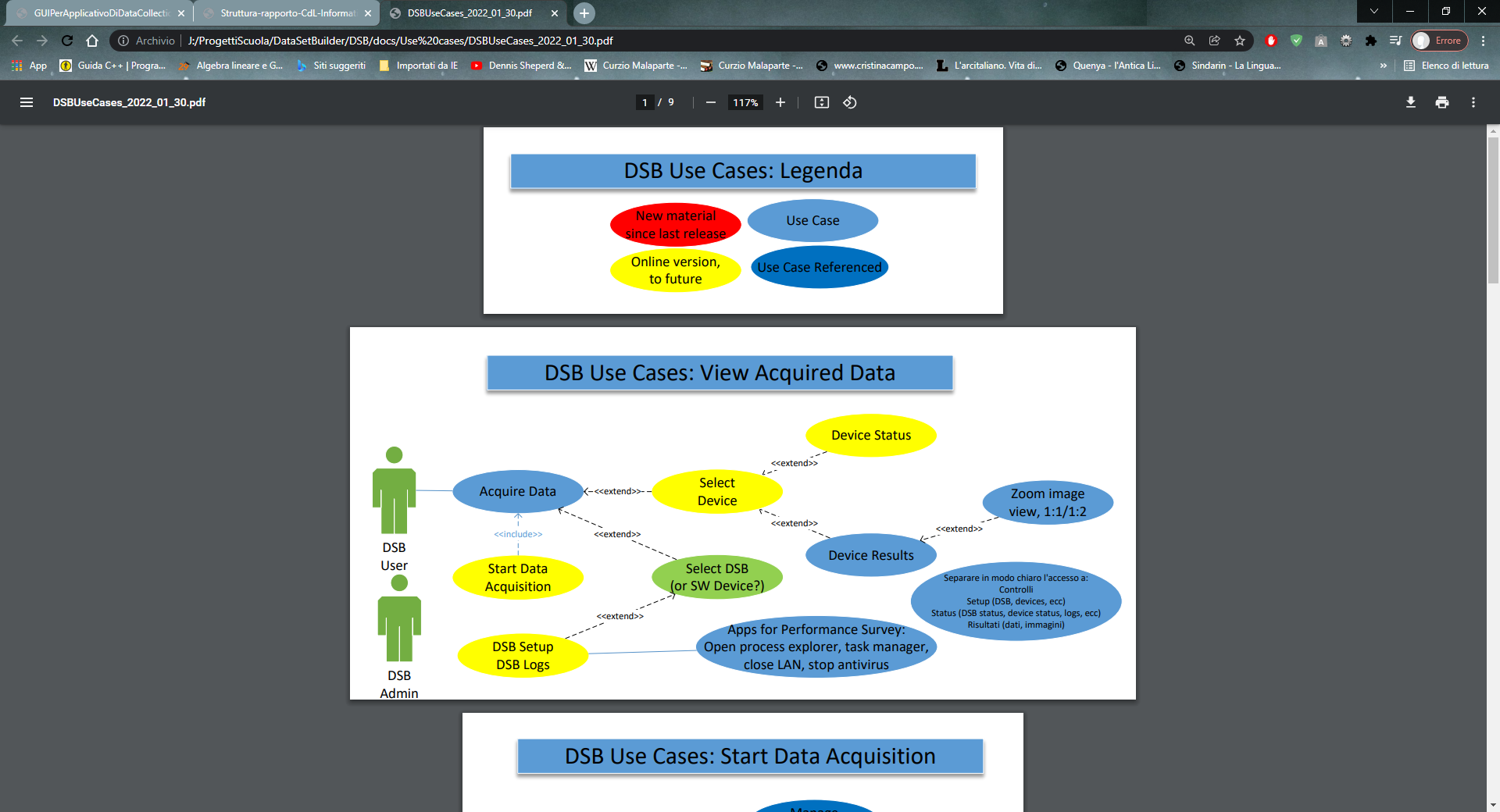


Figura 8: caso d'uso, view acquired data

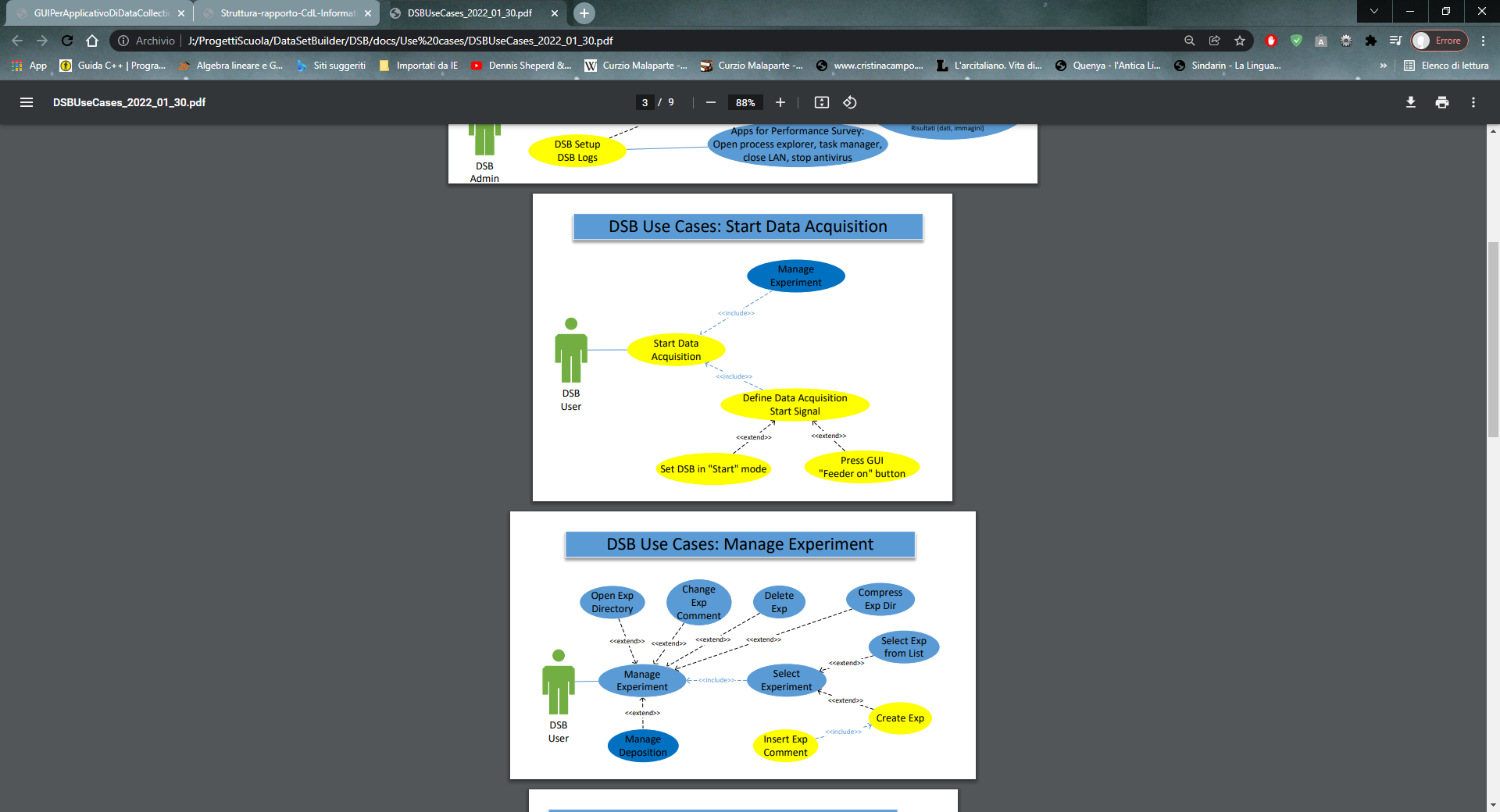


Figura 9: caso d'uso, start data acquisition

## Libreria grafica

Si descrive l’analisi preliminare al progetto che ha portato alla scelta di WPF (Windows Presentation Foundation) come libreria grafica del lavoro.

La prima decisione fondamentale del progetto ha riguardato la scelta della libreria grafica da impiegare. Le proposte iniziali erano le seguenti

* Windows Presentation Foundation, WPF
* Qt
* Windows Form
* Eventuali altre soluzioni…

Siccome il linguaggio di programmazione impiegato in questo lavoro di progetto è C# in ambiente .NET, Qt è stato scartato in quanto basato su C++ e poiché le altre librerie grafiche sono ugualmente adatte allo scopo.

Basandomi sulle esperienze del mio relatore, sembrava preferibile WPF poiché risultava maggiormente affidabile.

Entrambe le librerie sono più che affidabili, visto che sono presenti nell’ambiente .NET dal 2002, Windows Form, e 2006, WPF, e di entrambe è stata rilasciata di recente l’ultima versione stabile: 2020 per WinForms e 2021 per WPF.

È stato quindi necessario studiarne il confronto, basandosi anche sui riscontri della community di sviluppatori.

Il confronto fra le due librerie, fonte: <https://www.geeksforgeeks.org/difference-between-wpf-and-winforms/>

|  |  |
| --- | --- |
| **WPF** | **WinForms** |
| It is based on DirectX with XAML support | It provides access to the native windows library of common controls |
| It uses markup language for designing UI allowing the design of complex user interfaces | It does not use a markup language for design. In fact, it uses event-driven controls for the design |
| It can render fast as compared to WinForms, complexity, and support | It renders slow as compared to WPF |
| It can be used to develop and design both windows applications and web applications | It can only be used to develop and design windows applications. |
| It has unlimited UI customization and controls can be customized easily as it is totally written from scratch | In this, controls are limited and difficult to customize |
| It is easier to separate UI from back-end logic | It is tough to separate UI from back-end logic |
| It allows you to handle large data sets because of its in-built feature “user interface virtualization” | It does not have a “user interface virtualization” feature |
| It takes up more RAM than WinForms | It takes a smaller memory footprint |
| It is considered good when the application requires many media types, create a skinned UI, bind to XML, develop a desktop application having a web-like navigation style | It is considered good if you want to develop an application without much added modern functionality, more online resources |
| It offers effective and fully supported data binding as compared to WinForms | It offers data binding but in a limited way. Therefore, it’s less effective as compared to WPF |
| It also offers rich, interactive, animated, hardware accelerated, vector 2D and 3D capabilities for developing applications | It does not offer any rich, Interactive, animated, hardware accelerated, vector 2D and 3D capabilities as compared to WPF |
| It is a little bit tough, time-consuming, and complex to use WPF as compared to WinForms | It is simple to use WinForms as controls can be used easily, it’s less time-consuming and less tricky as compared to WPF |
| It has an inbuilt story boarding feature and animation model, it has the ability to use business objects in UI declaratively | It does not provide such features |

Tabella 1: comparazione WPF e WinForms

La scelta non deve essere basata unicamente su quale fra le due librerie sia la migliore paragonata all’altra, ma, e soprattutto, ponderata nell’ottica del progetto e di eventuali sviluppi futuri dello stesso.

Un esempio di questo è valutare il punto sull’impiego della RAM, soffermandosi su questa voce istintivamente si propenderebbe per WinForms, tuttavia dato l’ampio numero di dati raccolti negli esperimenti è molto probabile, e così è, che un applicativo debba a prescindere richiedere un grosso supporto di memoria alla RAM. Conseguentemente se WPF utilizza più memoria RAM rispetto a WinForms, non è rilevante in questo progetto specifico.

La decisione è ricaduta su WPF e i motivi sono i seguenti.

* Basato su XAML, un file XML  
  WPF si basa su due file: un file xml e un file cs. Il file xml è un file di markup al cui interno sono appunto scritti dei tag, logicamente analoghi ai tag in HTML, che permettono allo sviluppatore di inserire un elemento grafico. A livello grafico, inoltre, il cambiamento è immediato. Inserendo un tag di un Button all’interno dello xaml si ottiene il corrispettivo nell’interfaccia grafica. Conseguentemente è relativamente semplice sviluppare la presentazione di un applicativo con WPF. Il file cs invece contiene tutte le informazioni relative all’interfaccia. Quindi nominando il tag Button con una variabile, per esempio “MyButton”, è possibile chiamare su quell’oggetto i metodi specifici di un Button, incluse le proprietà quali le dimensioni, il colore e via discorrendo.
* È semplice separare la UI dalla logica back-end  
  Se l’architettura scelta per lo sviluppo è una Model View Controller questo punto è essenziale. Si desidera avere una libreria che favorisca il più possibile il lavoro anziché complicarlo, WPF mette in evidenza questa sua peculiarità rispetto a WinForms.  
  “Nel file .xaml sarà possibile disegnare l’interfaccia grafica, mentre il codice gestore della logica e degli eventi sarà implementato nel file.cs.”[[2]](#footnote-2) questa citazione conferma quanto scritto per i primi due punti presi in esame.
* Permette di gestire dataset di grandi dimensioni  
  Su questo punto, da parte del sottoscritto, è stata tralasciata un’analisi dell’importanza della virtualizzazione dell’interfaccia utente, ma la rilevanza in questa voce è la gestione di grandi quantità di dati. Si conosce già dal principio che il quantitativo di dati da gestire è alto e importante. Dunque è un punto di fondamentale importanza poiché atto all’obiettivo.
* Personalizzazione illimitata dell’interfaccia utente, controlli personalizzabili  
  Questa voce, sebbene non porti dei benefici nell’immediato presente al progetto, lascia aperti numerosi sentieri di personalizzazione futura. Questo applicativo potrebbe non essere limitato e dedicato unicamente a questo specifico lavoro o periodo di lavoro. È possibile che debba essere esteso, personalizzato, aggiornato, ecc…, ed è corretto pensare in quest’ottica. Rispettando anche quelli che sono i principi legati alla programmazione S.O.L.I.D.[[3]](#footnote-3), per un software aperto alle estensioni.

Sebbene WPF, così come sarebbe stato il discorso anche per WinForms se gli fosse stato preferito, possa sembrare datato e gli si sarebbero potute preferire altre librerie grafiche il suo supporto e la sua presenza sono stati garantiti anche in .NET 6[[4]](#footnote-4).



Figura 10: diagramma .NET 6

## Analisi di C# e WPF

Precedentemente al primo e al secondo incontro mi sono documentato sul linguaggio di C# tramite il libro di testo presente nella bibliografia di questo progetto: *Programmare con C#8 | Guida Completa.* Non era essenziale che io studiassi l’ultima versione del linguaggio, peraltro uscita nel corso del progetto[[5]](#footnote-5), poiché l’obiettivo era farmi un’idea globale del linguaggio.

Essendo un linguaggio orientato a oggetti, molto simile a Java, studiato durante i precedenti semestri alla SUPSI, l’approccio non è stato complicato.

Tuttavia alcune peculiarità del linguaggio le ho trovate molto utili e funzionali. Fra le quali cito brevemente:

* proprietà get/set, a seconda della posizione al momento della chiamata sarà interpretato come un getter o come un setter (int cm = persona.Altezza; 🡪 getter; persona.Altezza = 184; 🡪 setter)  
  private int altezza;  
  public int Altezza  
  {  
   get  
   {  
   return altezza;  
   }  
   set  
   {  
   altezza=value;  
   }  
  }
* la dichiarazione di una variabile locale “var” implicita, la quale deve essere subito assegnata durante la dichiarazione, il cui tipo di dato è deciso dal compilatore.
* LINQ, strumento in grado di interrogare un insieme di dati  
  “LINQ è un potente insieme di strumenti e funzionalità per scrivere con un’apposita nuova sintassi delle query di interrogazione di varie sorgenti di dati: collezioni di oggetti locali, dati XML letti da file oppure ottenuti da servizi web, database relazionali locali o remoti.”[[6]](#footnote-6)

Le prime due-tre settimane di progetto sono state investite sulla scelta della libreria grafica, ricaduta infine su WPF, e sulla presa di familiarità con il linguaggio C# e la libreria stessa. È stato quindi realizzato un primo programma, il classico “Hello World”, nel quale è stata inserita una parte di interfaccia grafica e un’ulteriore classe di C# oltre al controller dello xaml.

In quel programma di prova lo scopo era gestire degli eventi dell’interfaccia grafica e far partire un Thread che in modo casuale interrompeva un proprio counter interno mostrando a schermo un messaggio. Uno degli scopi era simulare dei possibili messaggi di errore.

Al termine di questa familiarizzazione si è lavorato sugli use cases, riassumendoli in un mockup che fornisse un’idea dell’interfaccia grafica e sull’implementazione software.

## Dati

# Tecnologie Utilizzate

Un breve riassunto sugli strumenti impiegati nello sviluppo e la loro versione.

## Visio e Pencil

Visio è uno strumento Microsoft impiegato per la creazione di diagrammi di ogni genere. In questo progetto è stato usato per la creazione dei casi d’uso.

Verisone usata: Microsoft® Visio® 2019 MSO (Versione 2112 Build 16.0.14729.20254) a 64 bit[[7]](#footnote-7).

Pencil è un software gratuito e open-source dedicato alla realizzazione di prototipi di interfacce grafiche. È stato conseguentemente impiegato per realizzare una demo grafica di quella che sarà l’interfaccia finale dell’applicativo.

Il risultato è presente nel capitolo “Mockup” di questo documento.

Versione usata: 3.1.0[[8]](#footnote-8).

## Visual Studio

Versione usata: Microsoft Visual Studio Enterprise 2019, Versione 16.11.8 con Microsoft .NET Framework, Versione 4.8.04084[[9]](#footnote-9).

## C#

Versione usata: C# 8.0[[10]](#footnote-10).

## Github e Github Desktop

Impiegato per la gestione del progetto: repository remoto, creazione/gestione di issues e creazione/gestione di Milestones. A un milestone è possibile assegnare una data di termine e un qualsiasi numero di issues. Hanno quindi rappresentato le scadenze da rispettare insieme alle rispettive issues.

L’applicativo per desktop è stato usato per impartire i comandi git necessari così come i commit. Ha favorito la gestione del progetto nella parte di git: commit, push e pull.

Versione usata: Version 2.9.6°(x64)[[11]](#footnote-11).

# Sviluppo

Nel capitolo sono trattati le parti rilevanti dello sviluppo dell’applicativo: prototipo grafico, gli user-control, le classi e i metodi degni di nota.

## Mockup

Il mockup fornisce un’immagine indicativa, poiché gli strumenti grafici fra Pencil e WPF differiscono nello stile,

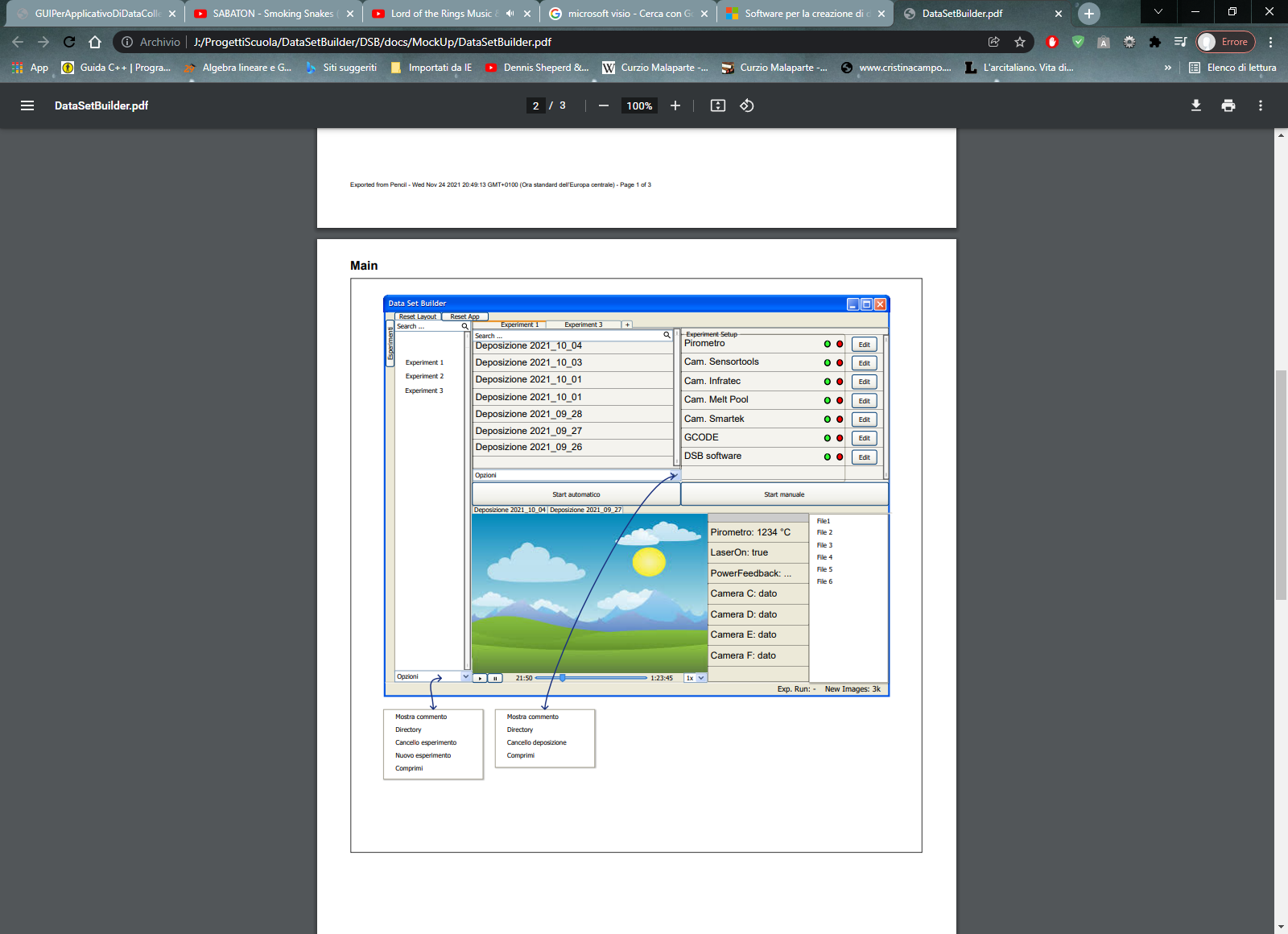


Figura 11: Mockup, presentazione finale del software

Per concepire il prototipo grafico mi sono basato sulla presentazione dei vari software applicativi che conosco e sulla richiesta di avere l’applicazione divisa in sezioni ben distinte: controlli, setup, elenchi, e via dicendo. Più i software usati come spunto sono rinomati, per esempio Visual Studio stesso, più è attendibile il fatto che gli sviluppatori abbiano seguito delle linee guida.

Quasi tutti gli applicativi di riferimento posseggono una sezione sulla sinistra con un elenco.

Visual studio possiede gli strumenti legati al file, per esempio l’elenco degli strumenti per WPF

Immagine che contiene testo, monitor, computer, interni

Descrizione generata automaticamente

Figura 12: Ispirazione mockup, Visual Studio

oppure l’albero gerarchico del progetto; nella figura si trova a destra, ma è mobile. Altri ambienti di sviluppo impiegati nel percorso accademico, come IntelliJ per Java, possiede questo albero sulla sinistra. L’esplora risorse di Windows sulla sinistra ha l’albero del file system.

Nel progetto è stato inserito sulla sinistra l’elenco degli esperimenti svolti e salvati sul file system. L’utente si aspetta di trovare sulla sinistra l’elemento gerarchicamente più in alto, quindi l’esperimento.

Esperimento > deposizione > dati

Inoltre, una volta aperti gli esperimenti da analizzare, non è necessario che l’elenco rimanga visibile. Il bottone sulla sinistra è stato progettato per chiudere il riquadro degli esperimenti. Per coerenza, le varie operazioni che si possono eseguire sulla lista sono nella stessa sezione: la barra di ricerca che filtra l’elenco secondo quanto digitato dall’utente posta in alto e le varie opzioni inserite nel menu in basso.

Un esperimento aperto occupa tutto il resto dell’applicativo e ogni esperimento aperto è inserito in una pagina la cui tab riporta il nome dell’esperimento aperto.

Per una logica di continuità e, ancora una volta, di coerenza, anche l’elenco delle deposizioni contenute nell’esperimento è costruito sul medesimo modello. Sopra l’elenco è posta la barra di ricerca e sotto di esso sono presenti le opzioni che si possono eseguire sulle voci.

La parte destra accanto all’elenco è dedicata alla lettura in tempo reale dello stato dei dispositivi, una sezione della parte online del progetto.

Le deposizioni si aprono nello spazio sottostante l’elenco mostrando nell’ordine le immagini con i relativi controlli, i corrispettivi dati misurati e il file explorer.

Il concetto dell’applicativo è basato sull’ordine di lettura cui noi siamo abituati: dall’alto al basso, da sinistra a destra.

Nel capito dedicato ai risultati è presente il manuale d’uso che spiega meglio quanto in questo capito è stato unicamente accennato.

Immagine che contiene tavolo

Descrizione generata automaticamente

Figura 13: Mockup, user-control relativo alla parte online

Sebbene non sia stata ancora implementata la parte online, le figure 13 e 14 ne mostrano i prototipi.

La figura 13 mostra la lettura in tempo reale dello status dei dispositivi. Il led di stato può segnalare errore, corretto funzionamento, e qualsiasi stato rilevante debba essere mostrato all’utente. È presente un riferimento all’attuale configurazione del dispositivo, oppure un altro riferimento relativo al dispositivo a seconda della necessità del team di lavoro, e un bottone che permette di modificare la configurazione aprendola con l’applicativo dedicato al dispositivo.

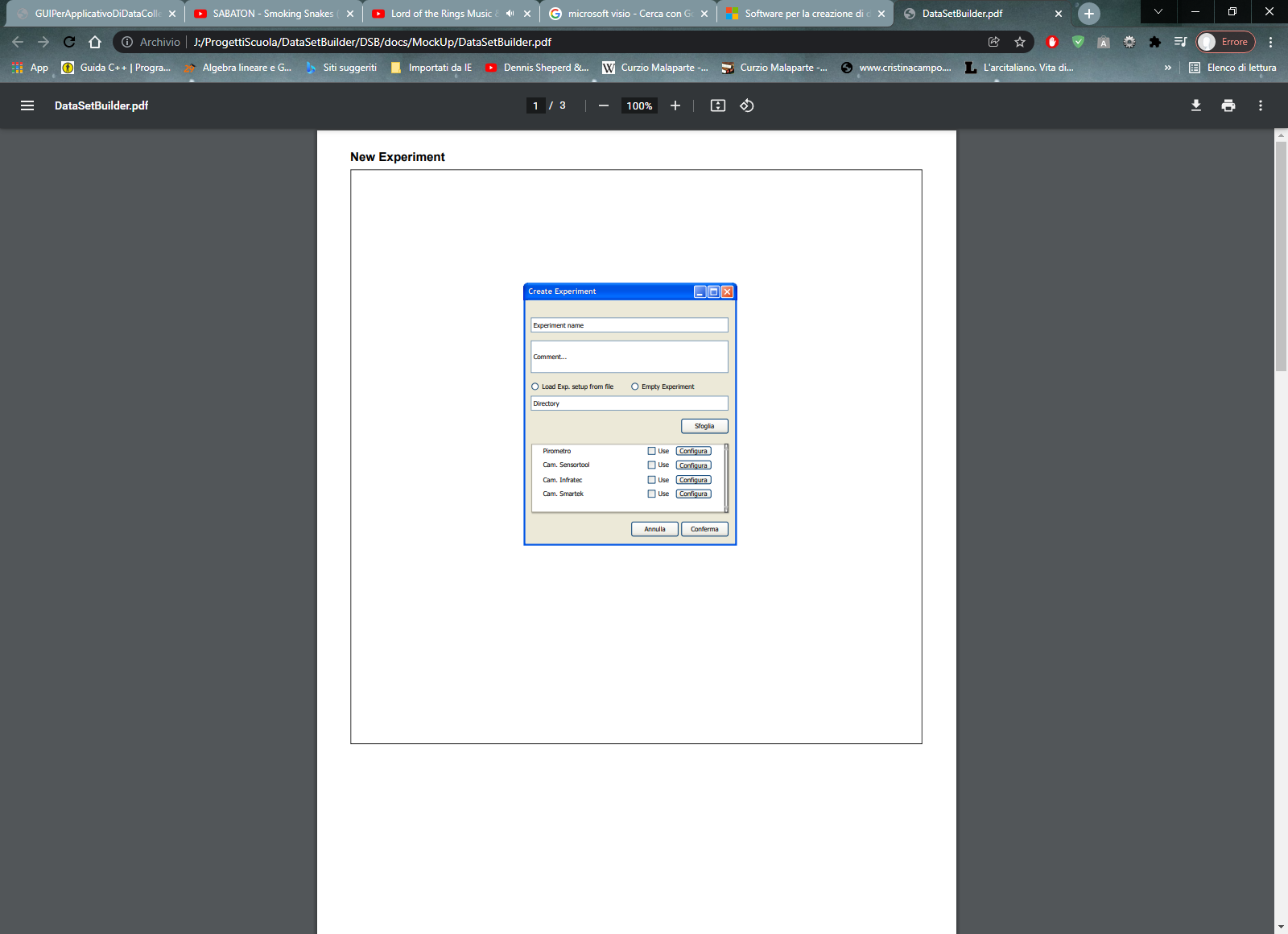


Figura 14: Mockup, parte online, creazione di un esperimento

La figura 14 mostra il concept per la finestra di creazione di un nuovo esperimento.

## User-control[[12]](#footnote-12)

Per un’interfaccia dinamica è necessario che i vari controlli e strumenti grafici siano istanziati al momento opportuno. L’utente non vorrebbe vedere uno slider per scorrere le immagini della deposizione prima che una deposizione sia aperta.

Conseguentemente bisogna ricorrere agli user-control.

La struttura di uno user-control ricorda la struttura della finestra principale: un file xaml e il suo relativo file cs per la logica.

Tuttavia la finestra principale non può essere istanziata più volte, mentre lo user control sì.

Lo user control permette di creare grafiche e controlli utilizzabili nelle varie porzioni dell’applicativo.

## Classi e metodi rilevanti

SPIEGAZIONE DI CLASSI E METODI RILEVANTI IMPIEGATI NEL PROGETTO

* diciotnary

# Analisi dei risultati

## Risultato

Immagine che contiene testo

Descrizione generata automaticamente

Figura 15: Ground Control, all’apertura

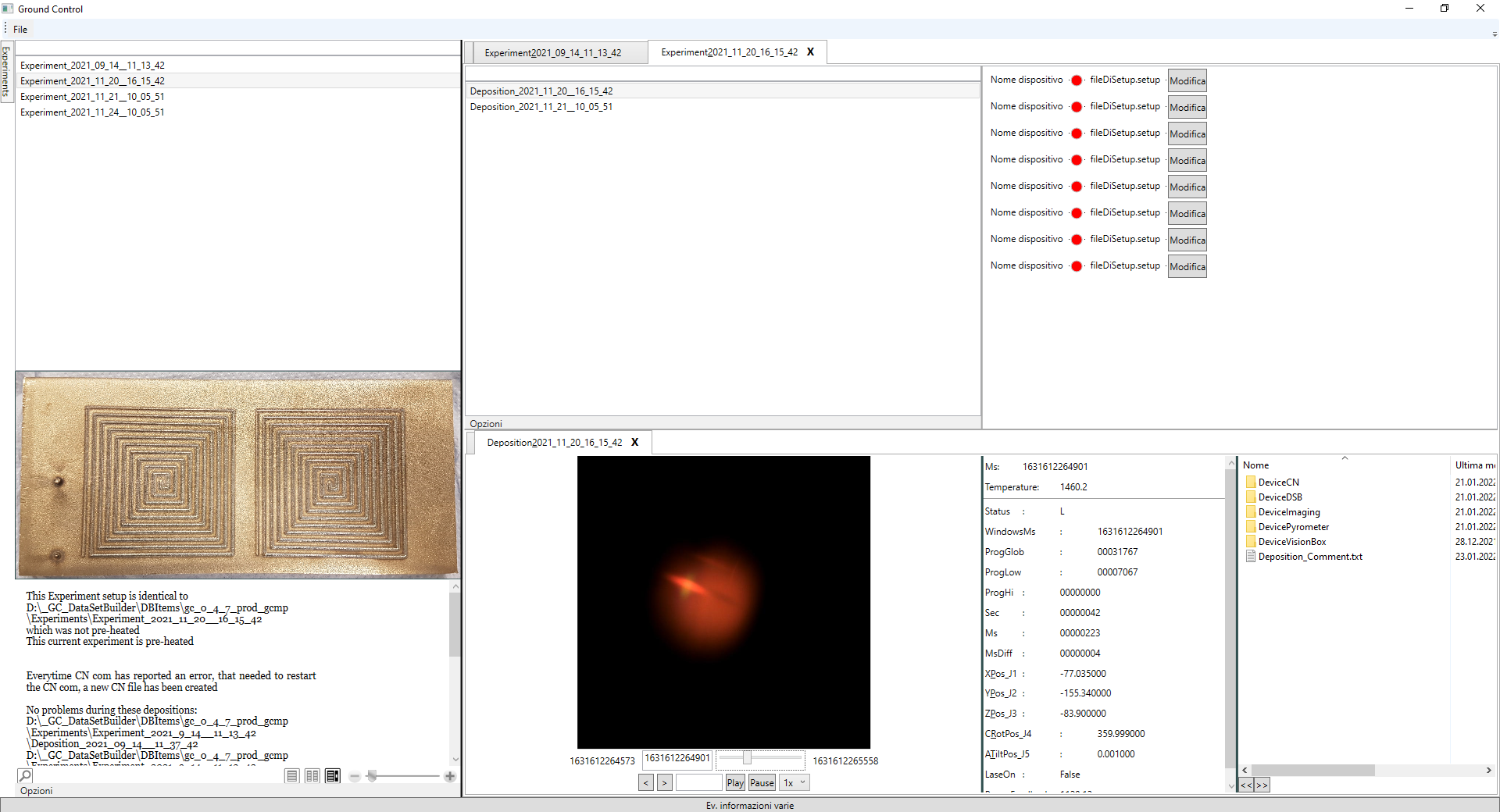


Figura 16: Ground Control, at work

## Manuale d’uso

In questo capito è spiegato come utilizzare l’applicativo Ground Control.

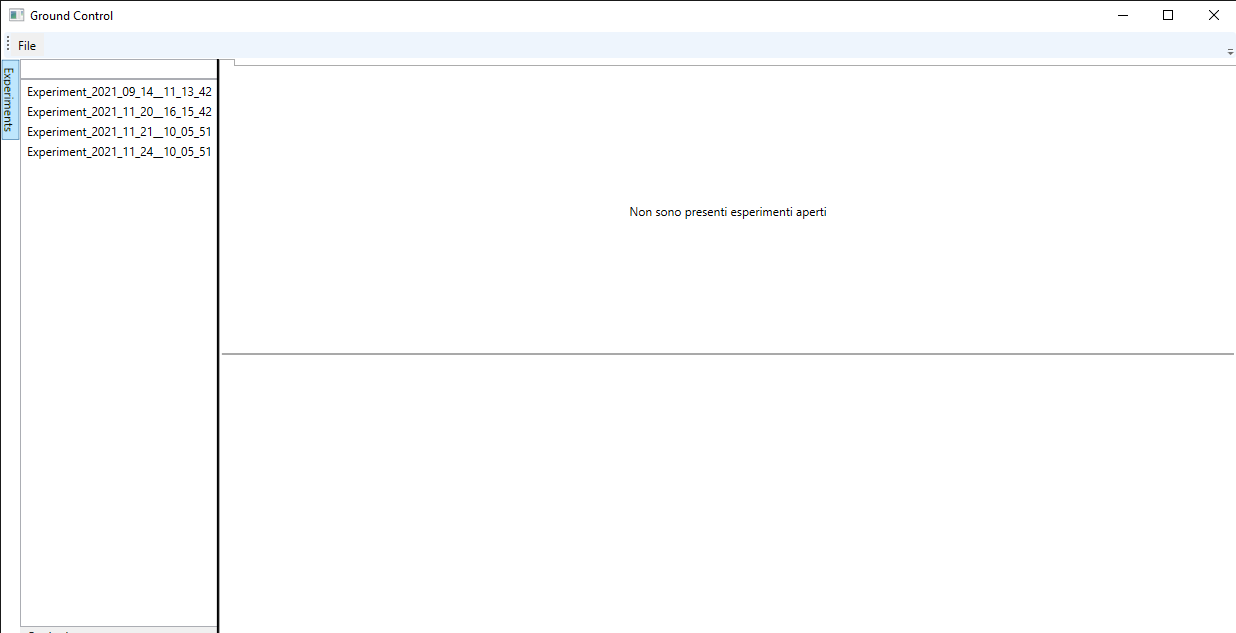


Figura 17: Manuale d'uso, GC aperto

Appena aperto, l’applicativo mostra l’elenco degli esperimenti presenti nel percorso configurato.

Il pulsante sulla sinistra del riquadro degli esperimenti permette di mostrare e nascondere la sezione.

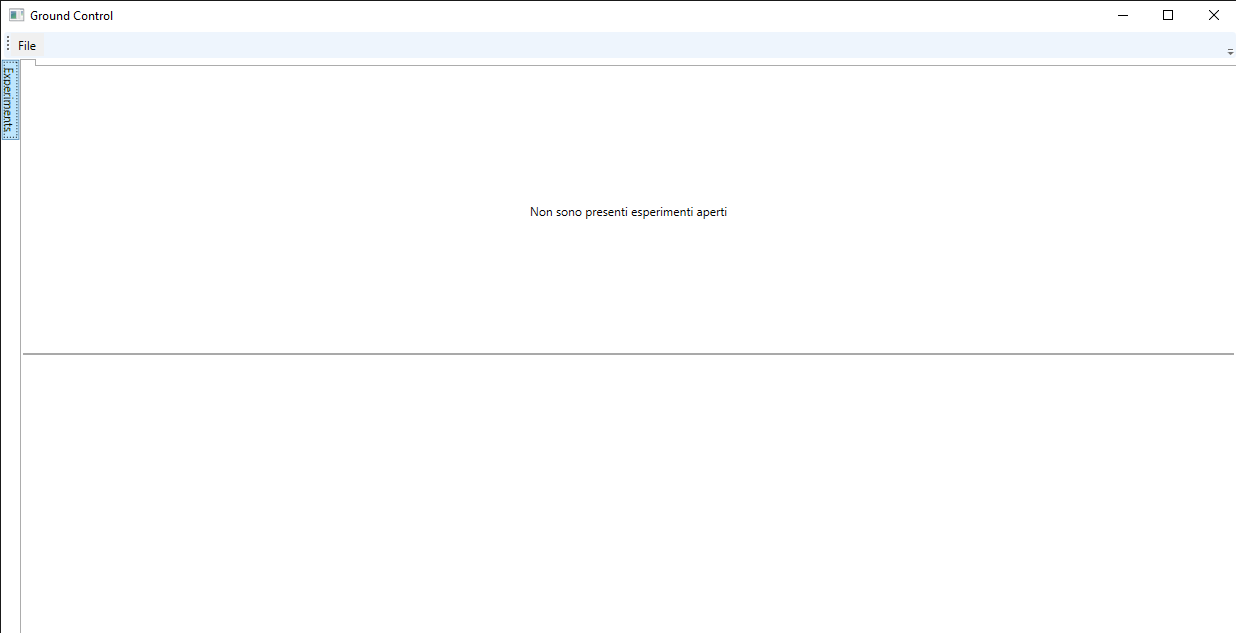


Figura 18: Manuale d'uso, elenco esperimenti chiuso

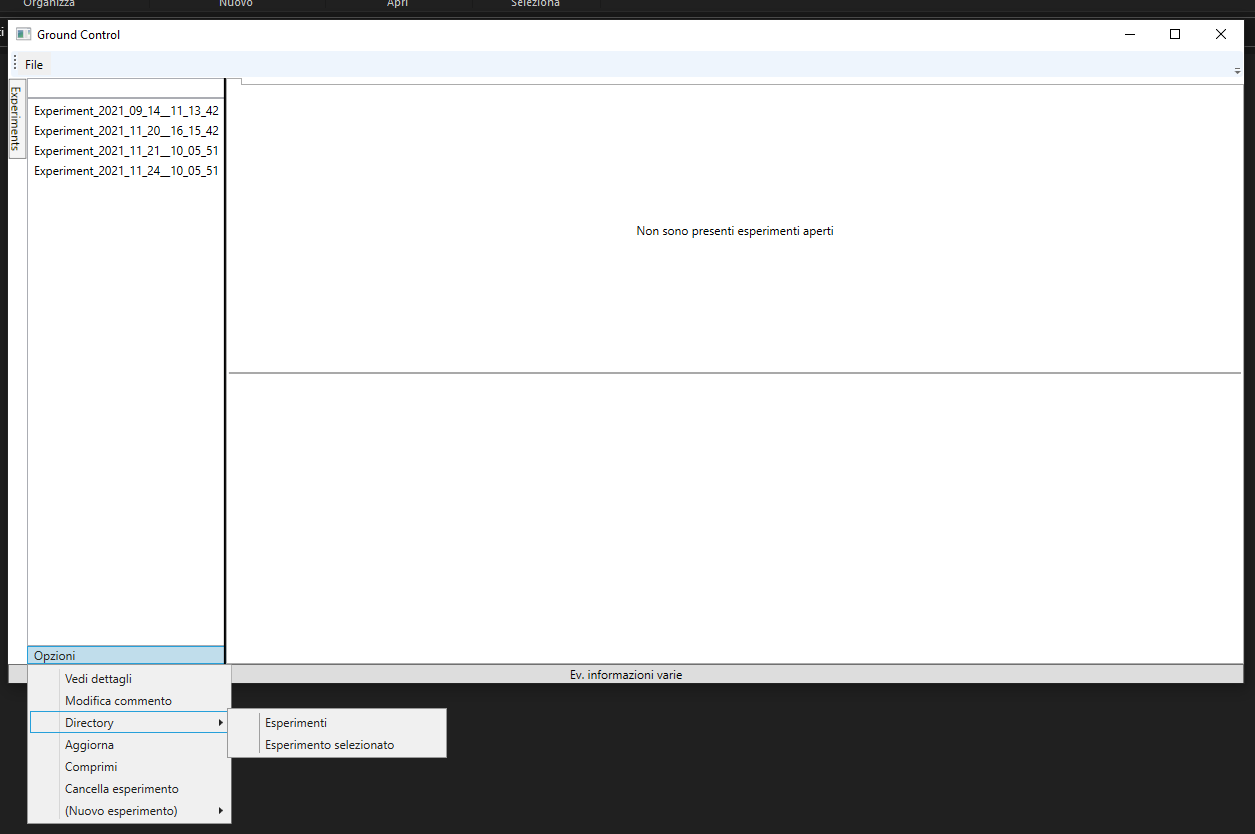


Figura 19: Manuale d'uso, opzioni esperimento

L’utente può eseguire le operazioni mostrate dalla figura 19.

* Vedi dettagli:  
  il menu mostra l’immagine di provino, se presente altrimenti ne mostra una predefinita (figura 20), e il commento testuale dell’esperimento. Entrambi si trovano nella cartella dell’esperimento. L’immagine si salva automaticamente, per il commento è necessario salvare il file con la parola chiave “Experiment” nel nome e che sia txt.
* Modifica commento  
  il menu apre il commento dell’esperimento con il programma predefinito della macchina per aprire i file di testo (es. Blocco note, Notepad++, eccetera).
* Directory  
  il menu permette di aprire nell’esplora risorse della macchina il percorso dell’esperimento selezionato oppure il percorso contenente gli esperimenti dell’elenco.
* Aggiorna  
  il menu aggiorna la lista degli esperimenti. Se vi sarà un esperimento in più o in meno all’interno del percorso configurato, il cambiamento sarà riflesso sull’elenco.
* Comprimi  
  il menu permette di comprimere l’esperimento selezionato in formato “zip”. Il file compresso si troverà nella medesima cartella dell’esperimento di origine.
* Cancella esperimento  
  il menu cancella la cartella dell’esperimento selezionato. Siccome è un’operazione importante, all’utente è chiesta una conferma prima di procedere con la cancellazione dell’esperimento.
* Nuovo esperimento  
  il menu consentirà di creare un esperimento nuovo vuoto oppure un esperimento nuovo basato su delle configurazioni esistenti. Facendo parte della parte online, questa voce è ancora da implementare.



Figura 20: Manuale d'uso, dettagli esperimento

Tramite l’opzione “Vedi dettagli” all’utente sono mostrati l’immagine provino e il commento dell’esperimento selezionato. Se, come nel caso della figura 20, l’immagine non fosse presente verrebbe mostrata quella predefinita.

L’utente può aggiungere un’immagine provino tramite la funzione Drag&Drop.

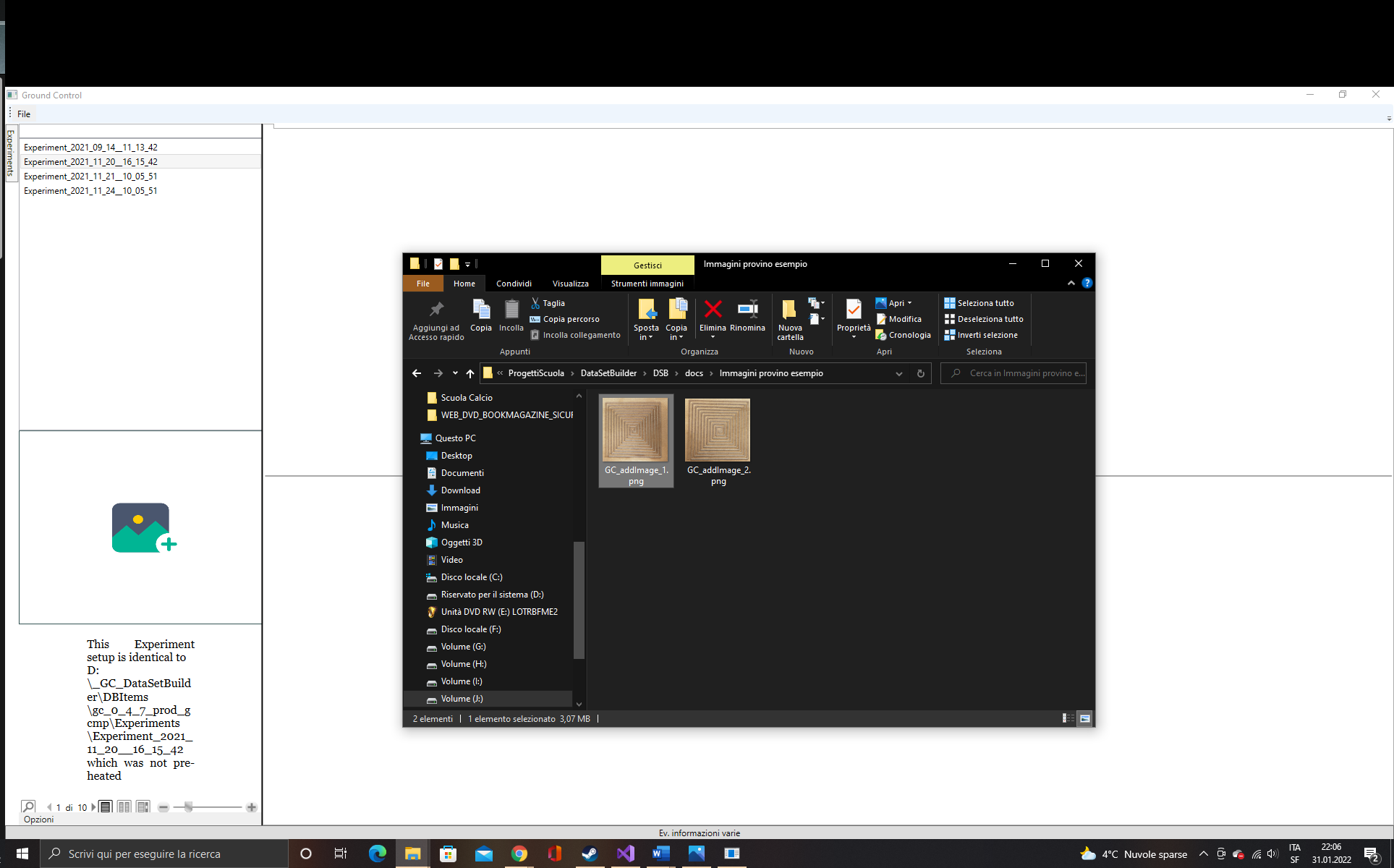


Figura 21: Manuale d'uso, Drag&Drop provino

Per aggiungere un’immagine di provino è sufficiente trascinarla sopra quella predefinita e rilasciare il mouse.

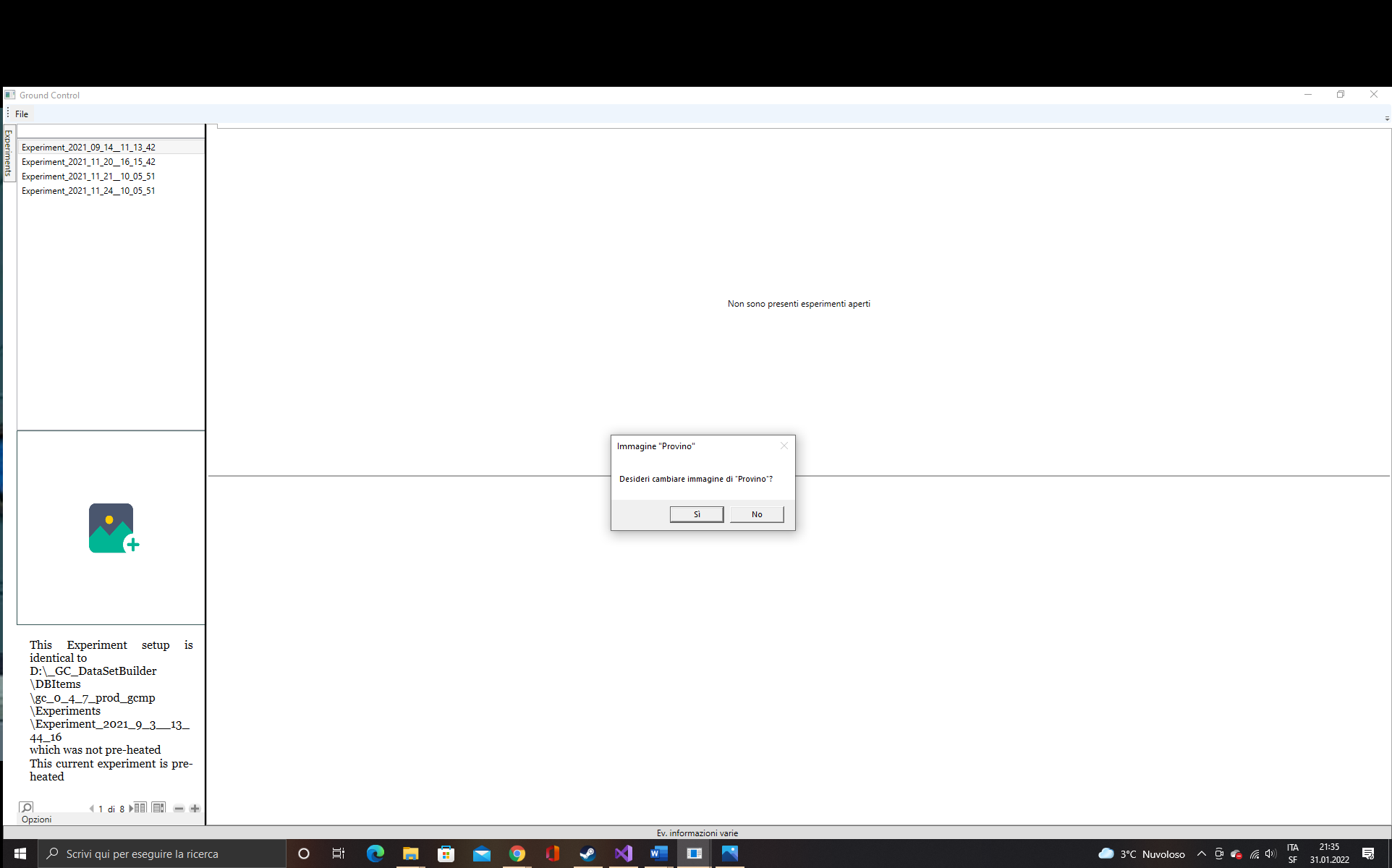


Figura 22: Manuale d'uso, conferma della nuova immagine provino

A questo punto viene chiesto all’utente se desidera cambiare l’immagine di provino dell’esperimento. Rispondendo in maniera affermativa l’immagine sarà salvata e mostrata all’utente.

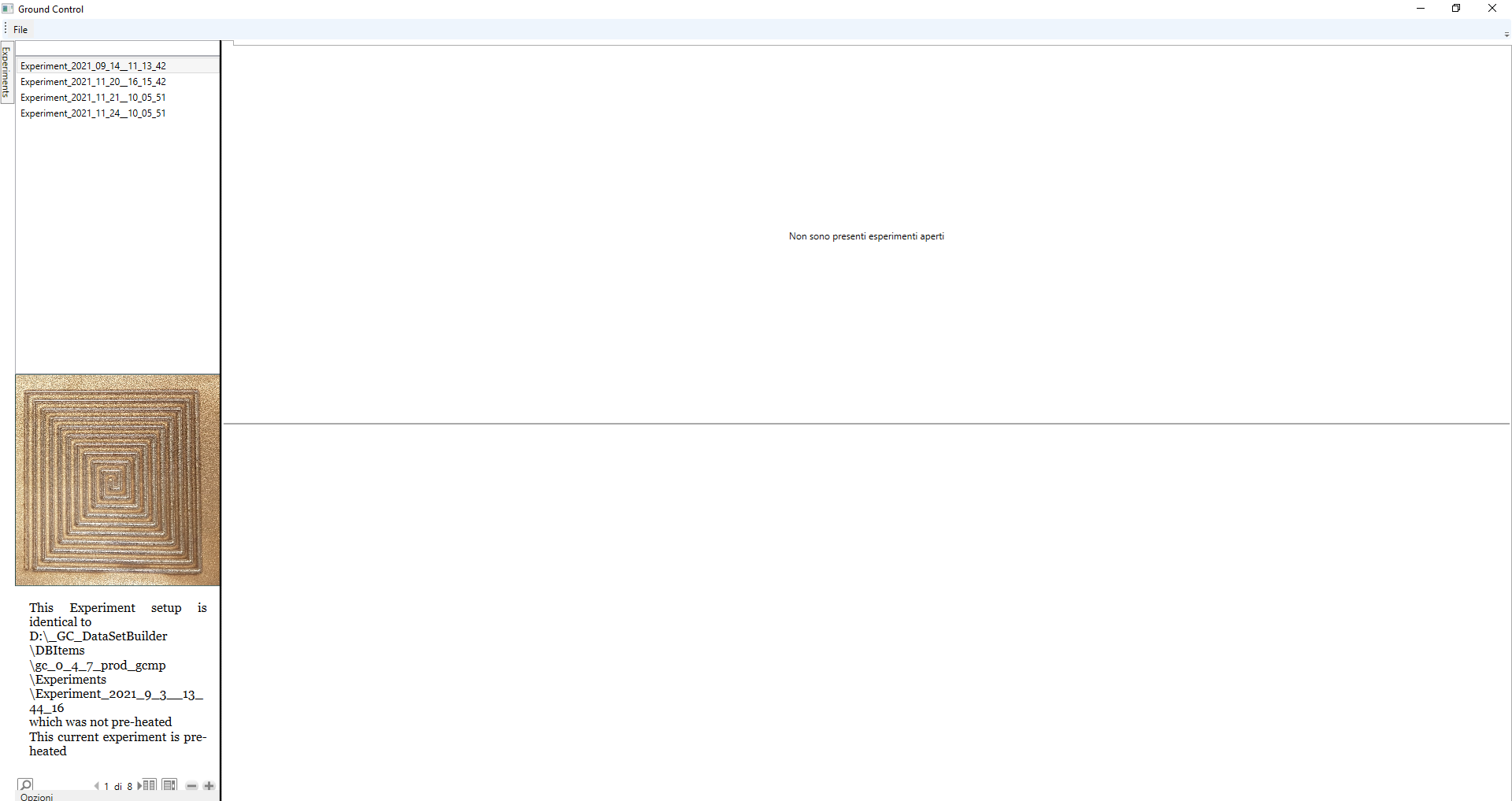


Figura 23: Manuale d'uso, nuova immagine provino

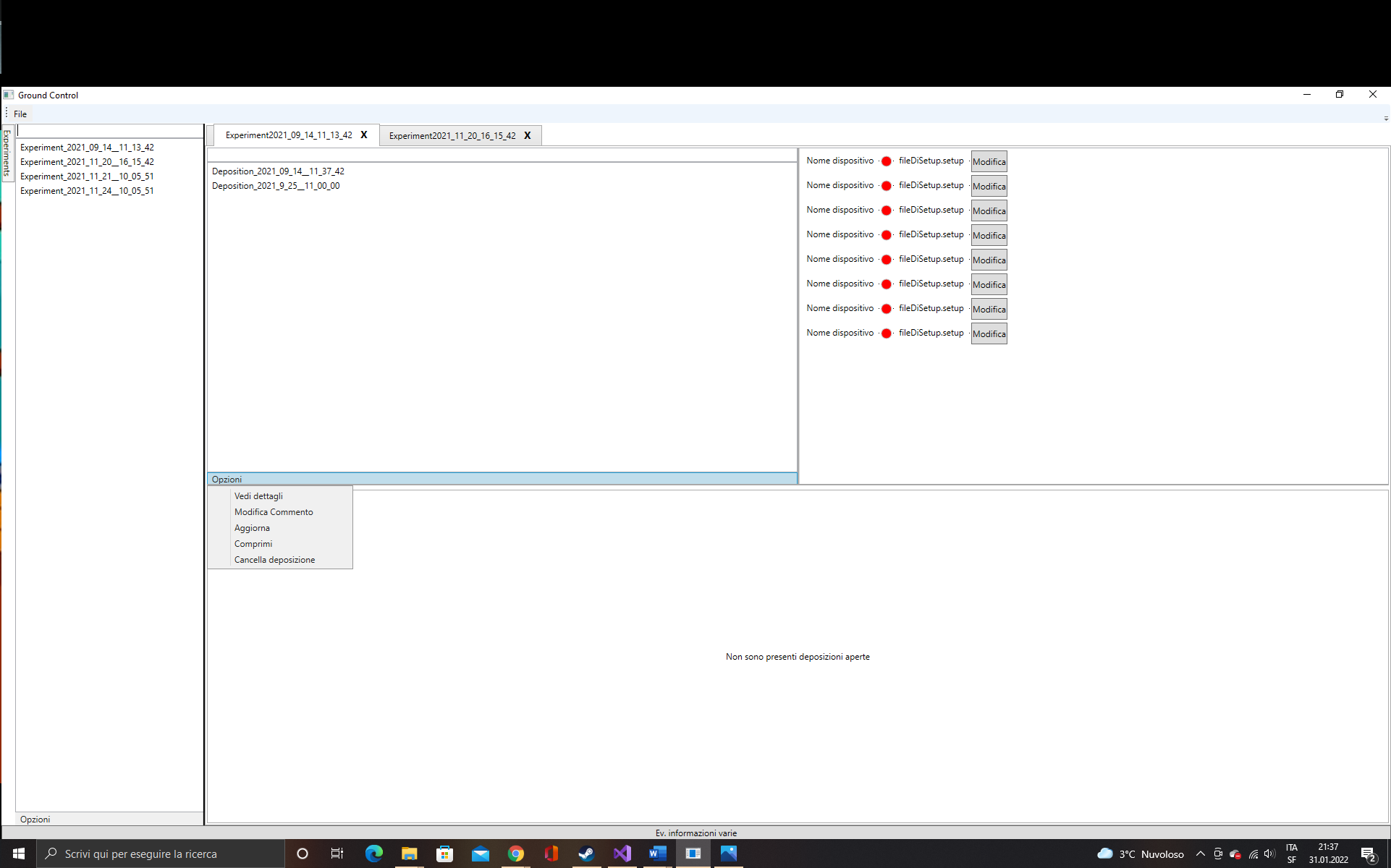


Figura 24: Manuale d'uso, opzioni deposizioni

Eseguendo un doppio click su un esperimento è possibile aprirlo nell’interfaccia in una nuova pagina con tab. Se ne recuperano le deposizioni disponibili in un elenco che viene mostrato e, idealmente con la versione online implementata e attiva, viene mostrato anche lo stato dei dispositivi connessi con la macchina.

Sotto il menu delle opzioni sono presenti le seguenti voci per le deposizioni.

* Vedi dettagli  
  In modo analogo agli esperimenti, anche per le deposizioni è possibile aggiungere un’immagine e avere un commento in formato testuale. Il file del commento deve contenere la parola chiave “Deposition” ed essere un txt.
* Modifica commento  
  Permette di aprire il commento della deposizione selezionata tramite l’applicativo predefinito.
* Aggiorna  
  Questa voce aggiorna l’elenco delle deposizioni.
* Comprimi  
  Permette di comprimere in formato “zip” la deposizione selezionata. Il file compresso si troverà nella medesima cartella della deposizione d’origine.
* Cancella deposizione  
  Il menu fornisce l’opportunità di cancellare una cartella di deposizione. Essendo un’operazione drastica, il software chiede conferma all’utente prima di concludere o meno la cancellazione.

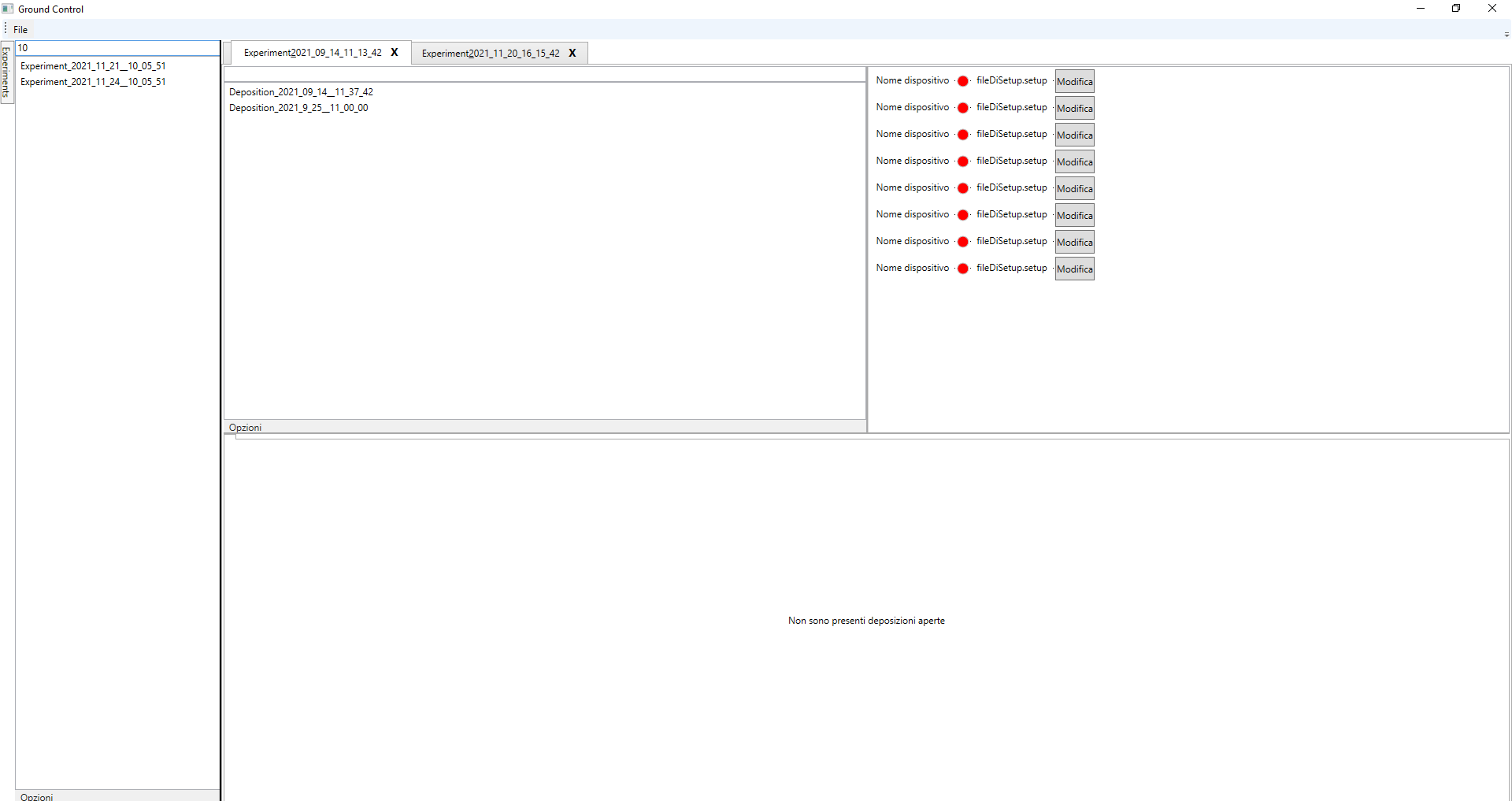


Figura : Manuale d'uso, barra di ricerca

Nelle rispettive barre di ricerca è possibile filtrare esperimenti e deposizioni in base al testo inserito. Non bisogna dare conferma dell’inserimento: l’applicativo reagisce a ogni modifica del campo.

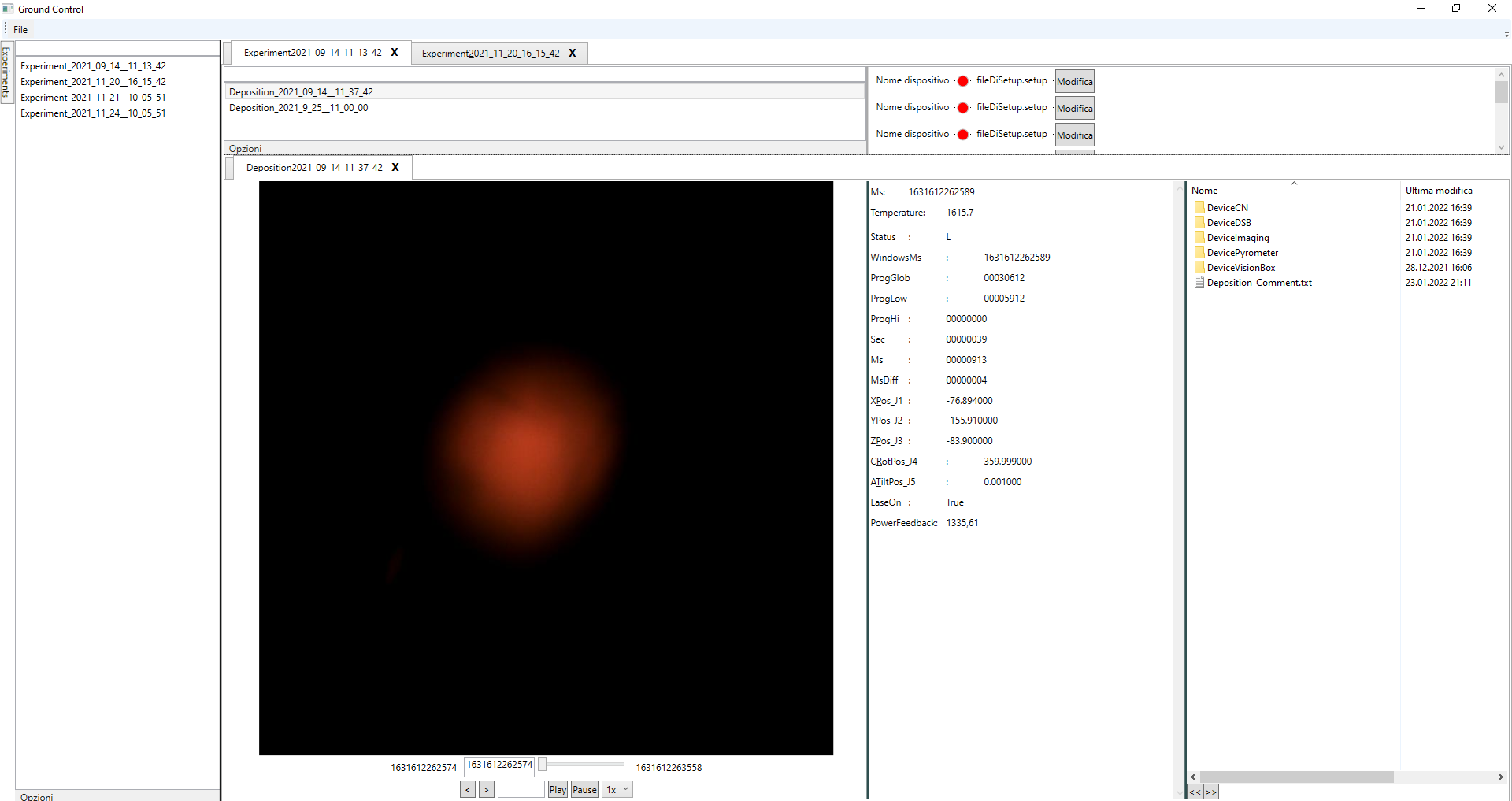


Figura : Manuale d'uso, deposizione aperta

Le diverse sezioni sono ridimensionabili, in modo che l’utente possa porre in evidenza ciò che desidera e ridurre la dimensione di ciò che non serve nell’immediato.

Nella figura 26 si nota che la sezione riguardante i dati della deposizione è stata ingrandita verso l’alto e potrebbe guadagnare ulteriore spazio se si chiudesse la sezione degli esperimenti tramite il bottone a sinistra.

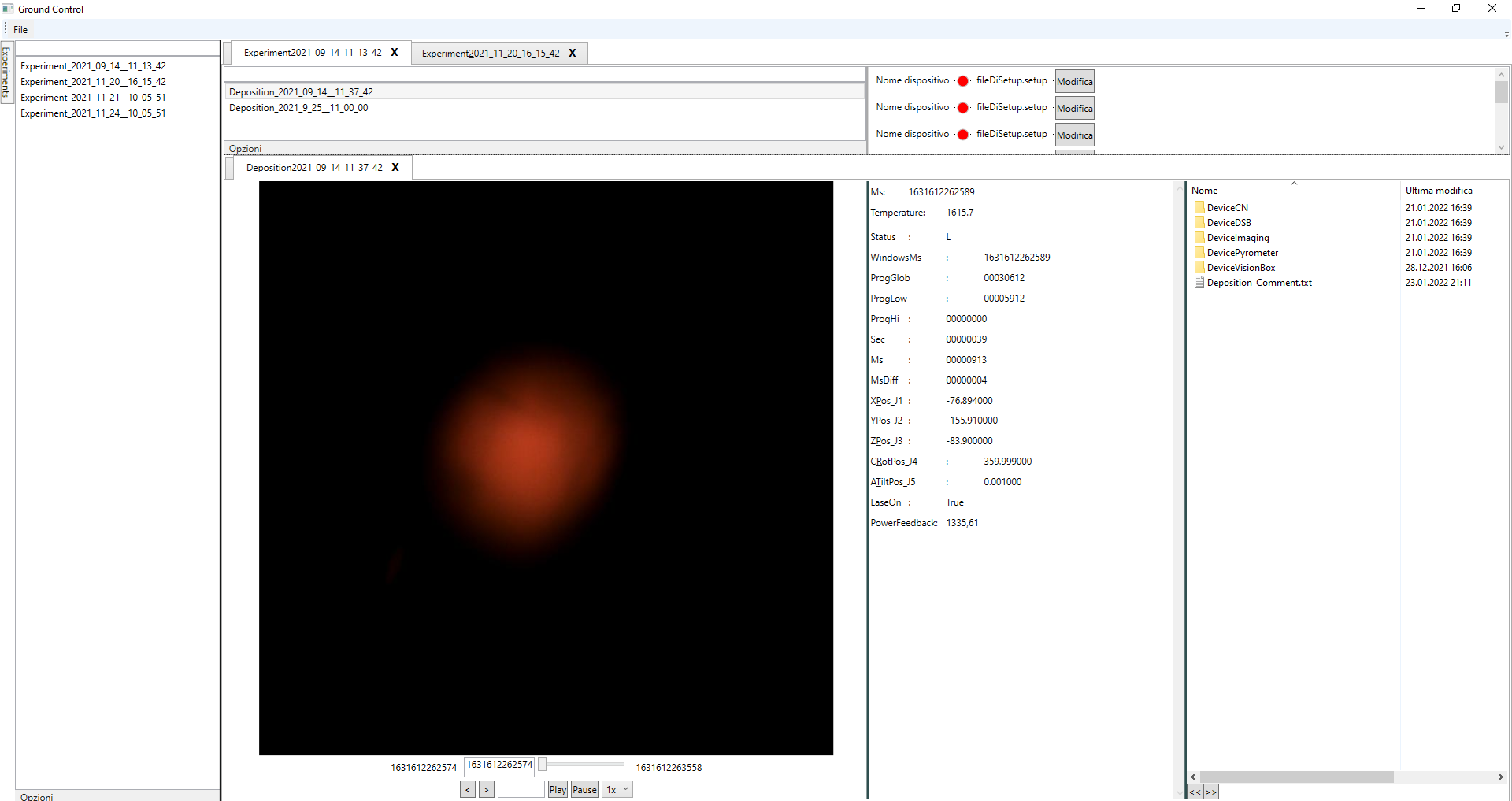


Figura 27: Manuale d'uso, controlli immagine

Al di sotto dell’immagine sono presenti alcuni controlli grafici. Nella parte superiore è presente uno slider in grado di scorrere le immagini, ognuna caratterizzata dall’istante di tempo in cui è stata salvata. Nel momento in cui si rilascia il cursore dopo averlo trascinato sarà mostrata l’immagine e i vari dati relativi all’istante di tempo mostrato dal riquadro sulla sinistra dello slider. Le etichette numeriche agli estremi della parte superiore indicano i due estremi dell’intervallo temporale.

La parte inferiore presenta due bottoni per scorrere avanti e indietro le immagini, un campo di ricerca nel quale indicare un istante di tempo, compreso fra i due estremi, e una volta confermato tramite il pulsante “Enter” dalla tastiera l’applicativo mostrerà l’immagine e i dati relativi al valore temporale ricercato, un pulsante di “Play” per riprodurre automaticamente un’immagine dietro l’altra, un pulsante di “Pause” per interrompere il precedente processo e un menu a tendina per selezionare la velocità di riproduzione delle immagini (1x una al secondo, 2x due al secondo, ecc…).

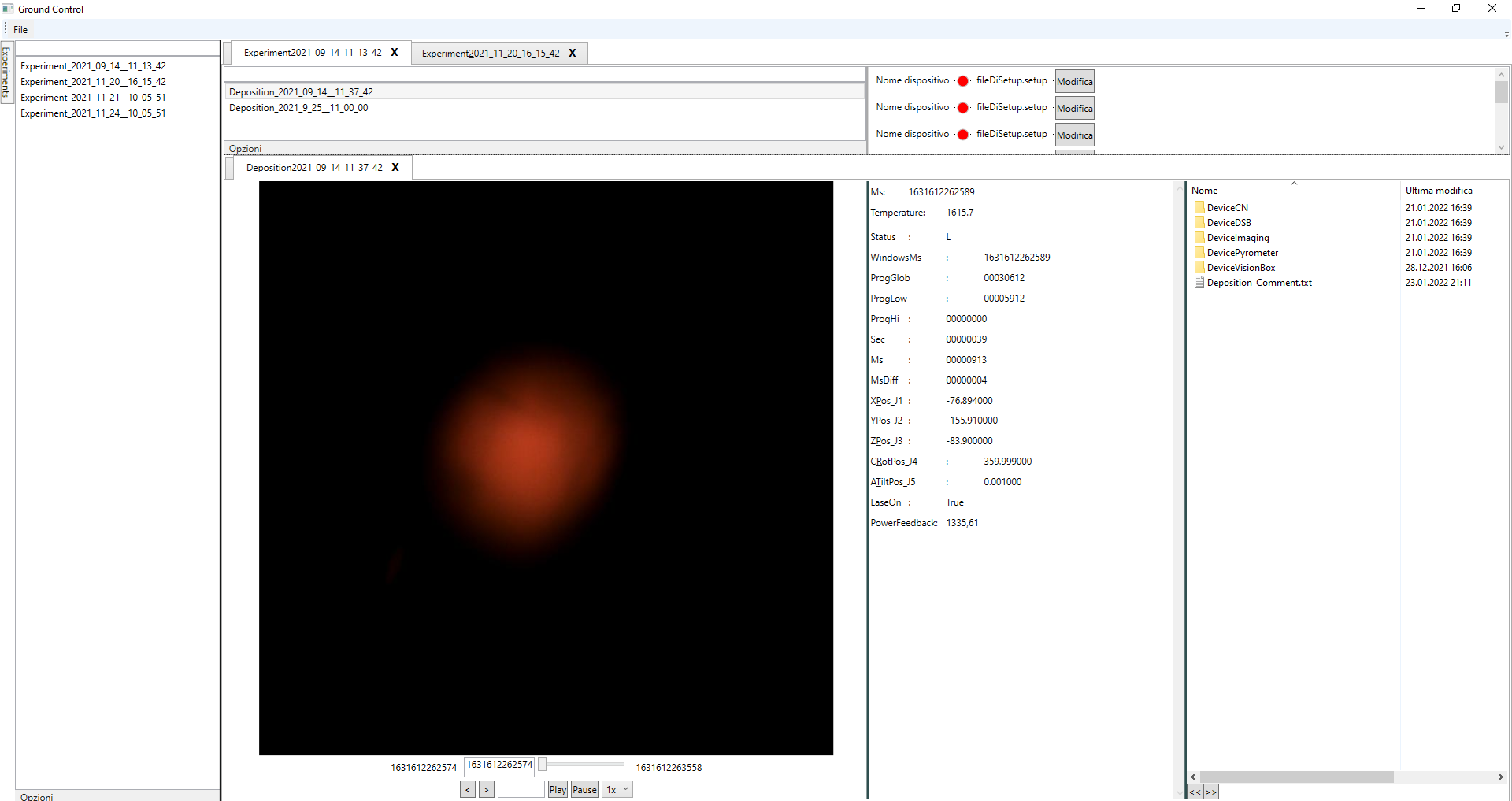


Figura 28: Manuale d'uso, dati e esplora risorse

Nei restanti riquadri l’utente può visualizzare i dati riguardanti un istante temporale, anch’esso indicato, e l’esplora risorse della deposizione aperta. I pulsanti in basso sono per navigare avanti e indietro nei file e nelle cartelle. Non è possibile navigare a un livello più alto della deposizione, poiché lo scopo è mostrare il contenuto della deposizione. Si comporta come il file explorer di Windows.

Immagine che contiene testo, computer, screenshot, monitor

Descrizione generata automaticamente

Figura 29: Manuale d'uso, menu File

In alto a destra nell’applicativo si trova il menu File. Al suo interno ci sono le seguenti funzioni.

* Percorso esperimenti  
  La voce permette all’utente di impostare un nuovo percorso radice per gli esperimenti, sia esso quello predefinito, nel quale caso sarà memorizzato come tale, sia esso uno temporaneo da usare fino alla chiusura dell’applicazione. In ogni caso gli esperimenti e le deposizioni aperte verranno chiuse, data la drastica operazione è chiesta la conferma all’utente prima di procedere.
* Esperimenti recenti  
  Passandovi sopra con il mouse sarà mostrata una lista con gli ultimi dieci esperimenti aperti dall’applicativo. Se riguardano il percorso corrente degli esperimenti i nomi saranno mostrati in nero, altrimenti in rosso. Solo gli esperimenti in colore nero si potranno aprire con la configurazione corrente del percorso del software.
* Riavvia  
  Si riavvia l’applicativo, prima di farlo è chiesta conferma all’utente.
* Chiudi

Si chiude l’applicativo, prima di farlo è chiesta conferma all’utente, così come è chiesta quando si desidera chiuedere l’applicativo dalla croce in alto a destra.

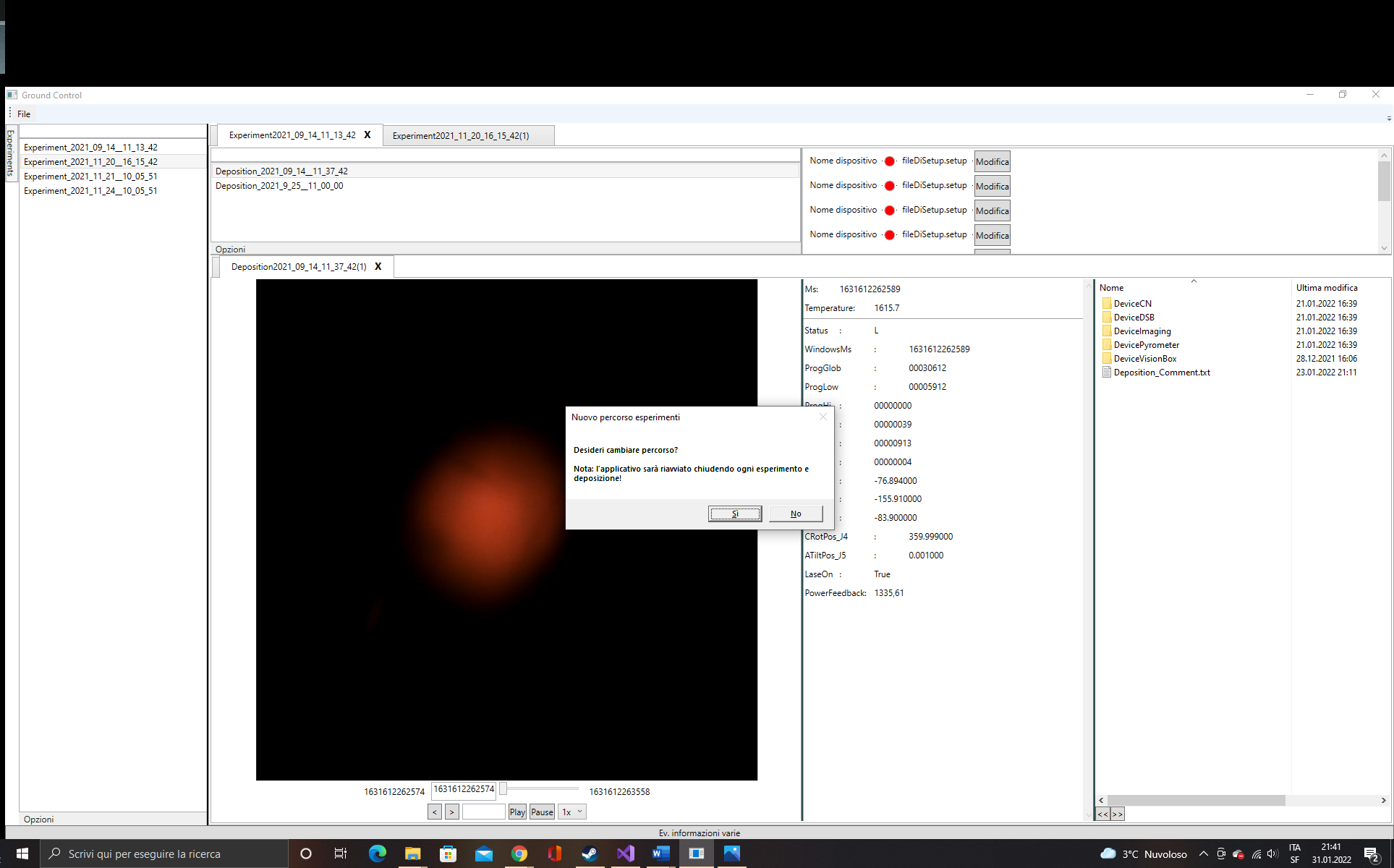


Figura 30: Manuale d'uso, conferma cambio percorso

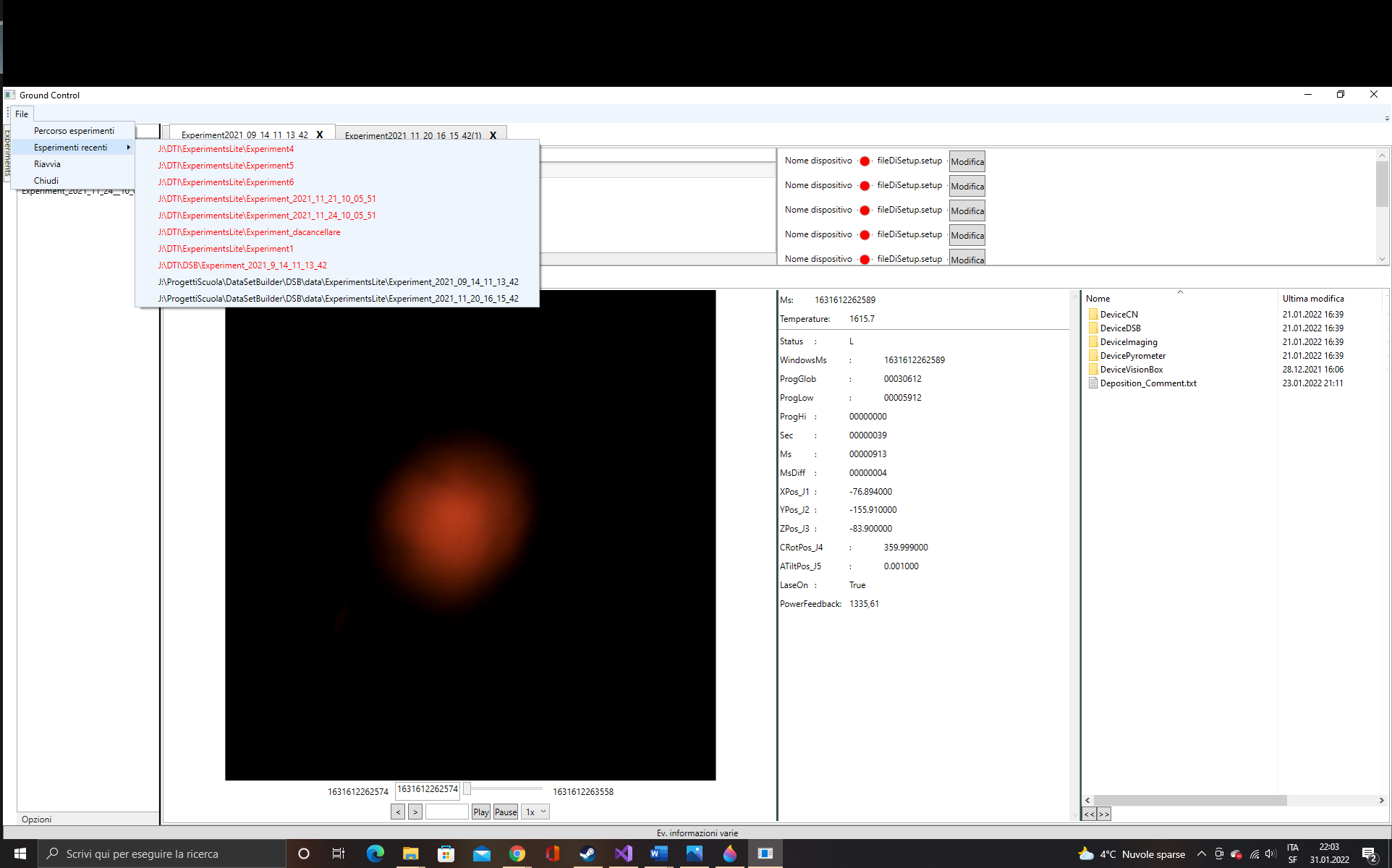


Figura 31: Manuale d'uso, esperimenti recenti

In rosso gli esperimenti aperti di recente, ma che si trovano in un percorso radice diverso da quello corrente. In nero quelli aperti con il percorso radice corrente.

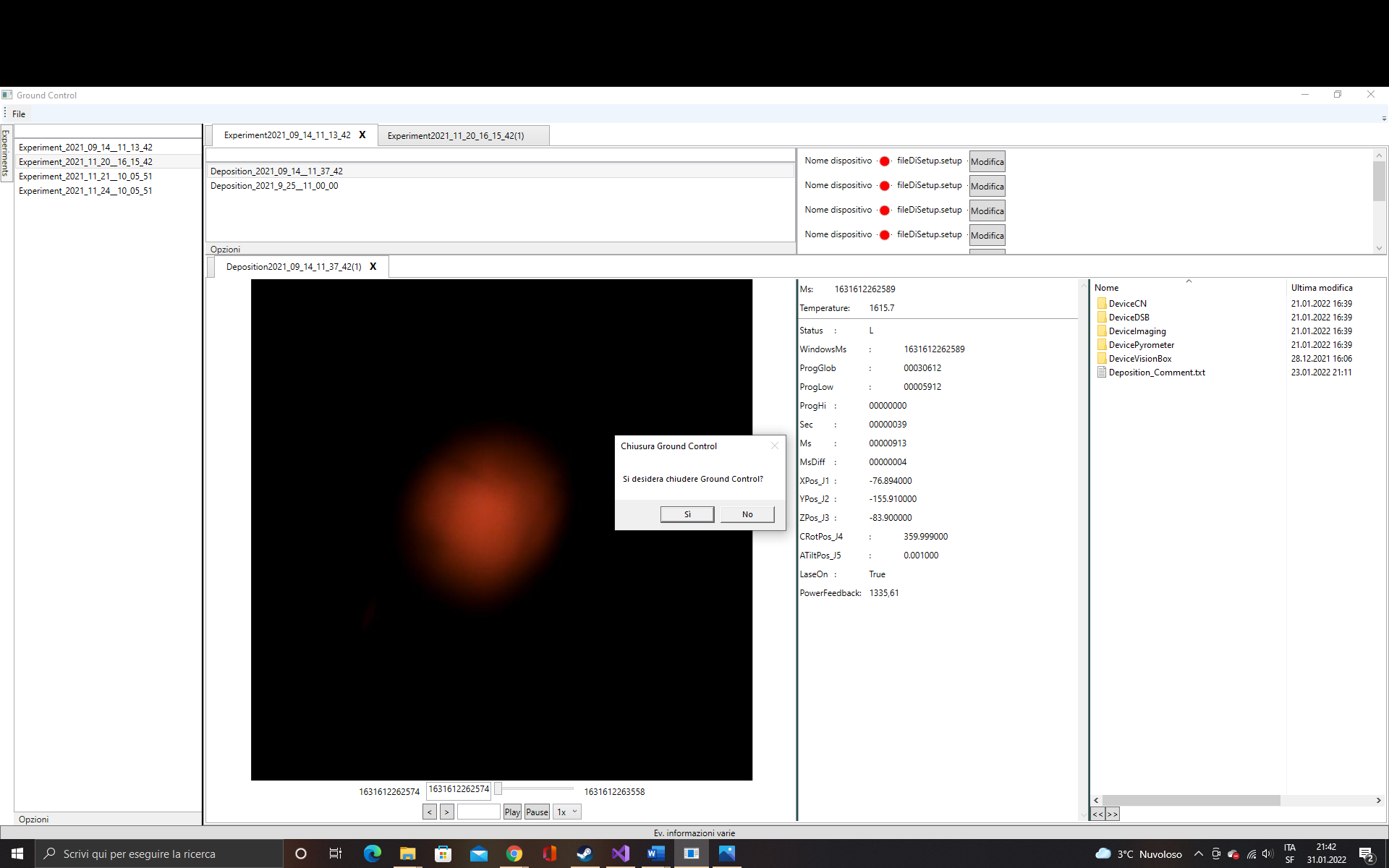


Figura 32: Manuale d'uso, conferma chiusura applicativo

## Problemi risolti

In questo sotto capitolo sono riportati alcune problematiche emerso in corso d’opera alle quali sono state trovate delle soluzioni.

### File Explorer

Come è mostrato nel mockup, una delle richieste è stata quella di avere una finestra di esplorazione delle risorse dedicata alla deposizione, ogni deposizione con la propria finestra, integrata nell’applicativo. Lo scopo è quello di mostrare tutti i file e cartelle contenuti nella deposizione aperta. Siccome la creazione di un esplora risorse occuperebbe diverso tempo, è stato preferito cercare ed eventualmente adattare qualcosa di esistente. Le ricerche sotto questo punto di vista non hanno portato risultati particolarmente proficui: le soluzioni trovate e provate erano inadatte, una inoltre faceva chiudere il programma poiché non reggeva il caricamento di tutte le immagini della deposizione. Quest’ultima è stata immediatamente scartata, poiché anche se nel corso del progetto si è lavorato con versioni più contenute delle deposizioni, nella realtà di quello che sarà l’impiego del software si lavorerà con deposizioni molto pesanti a livello di contenuti.

È stata infine trovata una soluzione già integrata in WPF. Secondo fonti in rete, confermate poi nell’implementazione, è possibile impiegare il controllo WebBrowser di WPF e usarlo come visualizzatore di documenti e risorse sul filesystem semplicemente impostando la source del controllo con il percorso desiderato. Nell’esempio studiato in rete sono integrati anche dei bottoni per poter navigare avanti e indietro nelle cartelle, se questo è possibile, in altre parole se si ha navigato in una cartella è possibile ritornarvi grazie al pulsante “back”. La sorgente di partenza è il contenuto della deposizione aperta.

//Nel webbrowser viene caricato il percorso come sorgente, affinché sia mostrato il contenuto del percorso come fosse un file explorer

this.FileBrowser.Source = new Uri(path);

//Evento collegato al click del bottone "<<" (indietro)

private void Previous\_Click(object sender, RoutedEventArgs e)

{

//Se la componente FileBrowser (WebBrowser) è in grado di navigare indietro allora lo fa (massimo fino al percorso specificato in fase di inizializzazione)

if (FileBrowser.CanGoBack)

{

FileBrowser.GoBack();

}

}

//Evento collegato al click del bottone ">>" (avanti)

private void Next\_Click(object sender, RoutedEventArgs e)

{

//Se la componente FileBrowser (WebBrowser) è in grado di navigare in avanti allora lo fa

//(unicamente se si ha già navigato in avanti rispetto al percorso base, es. entrare in una cartella)

if (FileBrowser.CanGoForward)

{

FileBrowser.GoForward();

}

}

### Binary Search

L’utente deve essere in grado di effettuare delle ricerche basandosi sul tempo, in ms, di quando è stato preso il dato. Premetto questa cosa: non ero ancora totalmente a conoscenza delle potenzialità di LINQ, quindi mi sono rifatto alle conoscenze pregresse sugli algoritmi di ricerca.

Sebbene ve ne siano anche di più performanti, la ricerca binaria converge verso il valore cercato, mentre altri algoritmi tendono a divergere sotto alcune particolari circostanze, proprie di ogni algoritmo.

È possibile eseguire la binray search sulle collezioni con il metodo BinarySearch(parametri)[[13]](#footnote-13). Tuttavia uno dei parametri di questo metodo è un Comparer. Ho preferito implementare un metodo Binary Search personalmente, principalmente poiché in caso di comportamenti errati sarei stato in grado di debuggare più facilmente il codice.

Infatti così è accaduto, poiché in dati casi ho dovuto eseguire delle migliorie e delle correzioni.

Il funzionamento di un algoritmo del genere è molto sistematico. Partendo dalla metà della lista ordinata, come in questo caso poiché le rilevazioni si svolgono nel tempo e sono incrementali, confronta il valore cercato con quello a metà lista. Se il valore cercato è più piccolo, l’algoritmo si sposta nella parte inferiore della lista, cerca la nuova mediana e prosegue con il confronto. Se il valore è più grande, l’algoritmo prosegue la ricerca nella metà superiore della lista cercandone la mediana e riprendendo il confronto.

Ovviamente se il valore non viene trovato bisognerà avvisare l’utente o, come nel caso di questo progetto, si cerca un valore prossimo. In questo specifico caso si cerca il valore inferiore più vicino. Un rischio legato al non ritrovamento del valore cercato è che l’algoritmo non si ferma. Data la natura ripetitiva, esso è implementato come funzione ricorsiva. Una funzione ricorsiva necessita una condizione di uscita. La più ovvia è quella del ritrovamento del valore cercato, ma le altre? In questo lavoro sono le seguenti.

Immagine che contiene testo, screenshot, monitor, nero

Descrizione generata automaticamente

Figura 33: BinarySearch, CNC

Le implementazioni sono identiche per ogni classe che si occupa delle ricerche: ImageSearcher, CNCSearcher e PyrometerSearcher.

Un controllo viene svolto se il valore che dovrebbe essere inferiore, left, diventa più grande di right. Questo significa che la nostra ricerca non è andata a buona fine e conseguentemente bisogna ricominciare cambiando il valore ricercato. L’offset è un valore impostato a -1 a ogni inizio ricerca e va a modificare il valore cercato di quel numero. Dopo 100 tentativi andati a vuoto, si esegue un salto con un offset moltiplicato per 7. I conteggi sono sempre resettati a ogni nuovo avvio della ricerca.

Il valore 7 è stato ricavato empiricamente in seguito a numerose prove. Questo poiché un valore cercato dal prof. relatore causò un’eccezione legata a un presunto loop infinito. Il valore è stato nuovamente provato dal sottoscritto più volte correggendo il moltiplicatore dell’offset, finché il valore 7 ha evitato l’eccezione.

Non ho svolto numerose ricerche a riguardo, ma talvolta i valori temporali delle misurazioni differiscono di cifre importanti rispetto al valore cercato e fra di loro e nel caso specifico questa differenza era di qualche migliaio, siccome nel caso il valore non fosse trovato lo si incrementerebbe tale dell’offset (-1) il metodo lavorava per molti cicli e a lungo.

È anche probabile che questo presunto loop infinito fosse dovuto a un altro mancato controllo.

Un ulteriore controllo riguarda il valore cercato. Se in un qualsiasi momento il valore cercato fosse più piccolo del valore minimo accettabile l’offset diventa positivo (1) e si ricomincia le ricerche cercando con valori sempre più grandi anziché sempre più piccoli.

Naturalmente se si ritrova il valore cercato si esce dalla ricorsione ritornando il valore.

L’ultimo controllo consiste nel confrontare il counter, che conteggia quante volte è stata eseguita la ricorsione per il valore attuale, con la variabile bigO. La variabile bigO rappresenta il valore O grande, il caso computazionale peggiore

con n che rappresenta il numero di elementi.

Inserendo il numero di elementi all’interno della formula si ricava bigO. In questo progetto bigO equivale a 23, che significa avere potenzialmente:

Potenzialmente 8 milioni di valori fra cui cercare. Al momento il numero di immagini massimo presente nelle cartelle delle deposizioni è 500'000 ca. Ground Control può accogliere ancora un numero maggiore di dati. Se dovesse crescere oltre quel limite sarà necessario ricalcolare bigO.

### Errore iniziale

Inizialmente la ricerca dei valori e delle immagini risultava molto lenta, ma questo non era dovuto alla metodologia impiegata, la binary search, ma bensì alla gestione I/O.

Per quanto concerne le immagini, il loro riferimento sono i nomi stessi salvati in una lista di stringhe, conseguentemente avviene un’operazione legata all’I/O a ogni ricerca. Per quanto concerneva invece le ricerche sui file del pirometro e del CN erano svolte nel modo errato. Esse erano eseguite sui file, quindi si avevano ripetuti accessi al file rallentando in maniera vistosa le operazioni. Bisogna anche ricordare che il problema è molto più palese con grandi dati, quindi non scalabile e non trascurabile; oltre che fondamentalmente errato.

Per ovviare a questo all’inizio della creazione dell’oggetto contenente i dati della deposizione, anziché memorizzare il nome del file, si salvano le righe di esso in una lista di stringhe e le ricerche si eseguono sulla lista.

### Automazione delle immagini mostrate

Una delle specifiche richieste consiste nell’avere le possibilità di mostrare le immagini in maniera automatica, senza quindi dover dare l’esplicito comando di cambiamento di immagine. L’utente ha inoltre la possibilità di stabilire il numero di immagini mostrate al secondo.

L’idea per l’implementazione era la seguente: creare un Thread, on un Task, cui delegare il compito di aggiornare automaticamente le immagini. Dopo averla realizzata veniva lanciata un’eccezione riguardante il Thread: esso non poteva avere accesso al controllo dell’immagine poiché questo è sotto il controllo di un altro Thread.

Di norma tutti i controlli legati all’interfaccia grafica sono gestiti dal Thread dell’interfaccia grafica, conseguentemente è necessario trovare un’altra soluzione. Sicuramente lasciare che se ne occupi il Thread grafico è impensabile poiché l’applicazione non sarebbe più utilizzabile.

In seguito a una ricerca e a un’analisi della situazione la soluzione è stata trovata tramite un metodo async[[14]](#footnote-15).

Struttura base di un metodo async:

Visibilità async void NomeMetodo()

{

await Metodo();

}

La parola chiave “await” consente al chiamante del metodo asincrono di riprendere la propria esecuzione. Al termine del metodo caratterizzato da await l’esecuzione riparte dall’istruzione successiva a tale metodo.

//Avvio della riproduzione automatica delle immagini

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

{

isAutomatic = true;

//Loop di aggiornamento dell'immagine

while(isAutomatic)

{

int ratio = getSpeed(); //velocità ricavata dal combobox

nextImage();

await Task.Delay(timeMs/ratio);

if (areImageEnd())

{

isAutomatic = false;

}

}

}

In questo modo il ciclo While non blocca l’interfaccia grafica poiché eseguito in modo asincrono e restituisce il controllo al chiamante.

### Chiusura delle tab

I TabItem nativi di WPF non posseggono la possibilità di essere chiusi mediante la classica croce presente per esempio nelle tab di un qualsiasi browser web.

Anche in questo caso è stato necessario svolgere una ricerca e un’analisi, fino al ritrovamento di una soluzione interessante e funzionante.

Essa comprende uno user-control, CloseableHeader, che contiene i controlli grafici necessari: la label per il titolo della tab e il bottone per la chiusura della tab.

Un’ulteriore classe, la CloseableTab, gestisce gli eventi sullo user-control eseguendo le operazioni su un’istanza di CloseableHeader che si istanzia come attributo. Queste classi sono state recuperate in rete già funzionanti.

Il codice del progetto è stato adattato poiché anziché istanziare un TabItem si istanzia un CloseableTab.

Osservando l’ereditarietà della classe si conferma quanto WPF sia personalizzabile

class CloseableTab : TabItem

inoltre è stato necessario modificare la proprietà Title aggiungendo il getter, eseguendo un cast a stringa

public string Title

{

set

{

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

}

get

{

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

}

}

Gli eventi della classe

Immagine che contiene testo, screenshot, monitor, schermo

Descrizione generata automaticamente

Figura 34: CloseableTab, eventi

concernono la visibilità del bottone di chiusura, unicamente quando la tab è selezionata, il colore della croce sul bottone quando ci si passa sopra con il mouse e la chiusura della tab al click sul bottone rimuovendola dal TabControl, parent dell’oggetto della classe.

Chiaramente qui sorge un problema. La tab si chiude, ma è necessario anche rimuoverla dal dizionario, altrimenti non si potrebbe più riaprire lo stesso esperimento o la stessa deposizione. Si deve trovare una meccanica affinché si rimuovano le tab anche dai rispettivi dizionari di controllo. Siccome bisogna anche soddisfare un’altra richiesta, quella che il medesimo esperimento o la medesima deposizione possano essere aperti più volte, il problema della rimozione della tab dai dizionari si risolve implementando questa richiesta. Ogniqualvolta si apre una deposizione o un esperimento si verifica che sia presente nel dizionario, nel caso lo fosse si aggiunge un postfisso simile a quello che possiedono i file o le cartelle copiate.

* Experiment\_1234
* Experiment\_1234(1)

Così facendo è possibile chiudere le tab a piacimento poiché i risultati possono essere aperti in più copie.

Ovviamente per recuperare i dati bisogna troncare la parte di copia del nome del risultato. Sebbene siano in tab differenti, i due esperimenti dell’esempio sopra si riferiscono ai medesimi dati nel medesimo percorso dettato dal nome del risultato.

### Refactoring limitato sullo user-control

È stata riscontrata la seguente “problematica” sul refactoring degli user control. Quando si rinomina il file xaml, si rinomina anche in modo automatico il suo cs, tuttavia Visual Studio segnala in seguito degli errori riguardanti un file non trovato. Si tratta proprio dello xaml con riportato il vecchio nome. Indagando all’interno dello xaml si nota che i riferimenti non sono stati rinominati, conseguentemente l’ambiente di sviluppo rivela l’errore.

x:Class="DataSetBuilder.user\_controls.DepoItemBody"

Qui sopra è riportato uno dei riferimenti che non si rinomina in maniera automatica.

### Impossibilità di separare fisicamente file xaml e relativo cs

Sebbene siano logicamente separati, a livello di progetto è impossibile separare fisicamente, in due cartelle diverse, i due file.

Immagine che contiene testo, screenshot, monitor, interni

Descrizione generata automaticamente

Figura 35: user control, albero del progetto

Come si nota dalla figura 15 il file cs è gerarchicamente sottostante e contenuto nel relativo file xaml.

## Sviluppi futuri

Sono ora elencati i possibili sviluppi e accorgimenti futuri riguardanti il progetto.

### Parte online e collegamento al database

L’integrazione del database e conseguentemente della parte online dell’applicativo sarebbe il passo successivo. L’utente è in grado di creare un esperimento copiando i setup esistenti dei dispositivi o ricreandoli partendo da zero. Siccome la parte offline funziona come applicativo di consulta dei risultati è necessario rendere la parte online agibile e manipolabile unicamente quando ci si trova sulla macchina in grado di far partire gli esperimenti e raccoglierne i dati.

### Gestione del click sullo slider

Attualmente lo slider funziona trascinando il cursore e al rilascio il sistema si occupa delle ricerche dell’immagine e dei valori. Tuttavia se l’utente clicca sullo slider, il cursore si sposta ma non avviene la ricerca, a meno che non si clicchi nuovamente sulla stesso punto. In quel caso l’interfaccia grafica sembrerebbe rilevare un piccolo trascinamento del cursore e quindi mostra i risultati.

La risoluzione più ovvia e immediata di questa difficoltà sarebbe quella di inserire un evento sullo slider che rileva il cambiamento di valore di esso, ma emergerebbe un problema nel momento in cui l’utente desidera trascinare il cursore. L’evento sul cursore si attiva appunto quando termina il trascinamento ed esso viene rilasciato, ma il cambiamento di valore dello slider avviene continuamente. Ciò che per l’utente è una questione di pochi millesimi di secondo e di pochi centimetri per il sistema potrebbe risultare un cambiamento di 200-300'000 valori. Sempre che i valori nelle deposizioni future siano sempre questi in media, ma se aumentassero arrivando a 1'000'000 significherebbe, nel caso peggiore, di chiedere al sistema di effettuare un milione di ricerche e mostrarne i risultati in qualche ms.

È stata testata questa soluzione e tali problemi sono emersi. Tuttavia avendo l’alternativa dello scorrimento del cursore non è stata cercata un’alternativa durante lo sviluppo di questo progetto.

### I Dictionary

Non avendo ancora gestita la rimozione degli elementi dai dizionari quando le tab vengono chiuse dall’utente con il mouse è verosimile che i dizionari possano crescere in maniera significativa se l’applicativo fosse sempre attivo nel tempo senza alcun riavvio. Non è chiaro cosa possa comportare un aumento significativo degli elementi nei dizionari nell’applicativo poiché questa situazione non è stata provata.

Sarebbe opportuno studiare la rimozione degli elementi quando essi vengono chiusi a livello di interfaccia grafica.

### Spostamento tab

A seguito dell’implementazione della CloseableTab le tab degli esperimenti e delle deposizioni non sono più spostabili graficamente come è possibile fare in un qualsiasi browser web. La causa è con ogni probabilità dovuta alla classe citata, poiché gli eventi legati allo spostamento delle tab sono legati alle istanze di TabItem.

Sono state provate delle piccole modifiche al codice, nel particolare la sostituzione di TabItem con CloseableTab nei metodi citati, ma non è stato riscontrato alcun cambiamento.

### File di configurazione per i device

È verosimile che nel corso degli esperimenti vi possano essere dei cambiamenti a livello di device utilizzati. L’utente dovrebbe avere la possibilità di configurare nuovi dispositivi e/o rimuovere e manipolare quelli esistenti all’interno di Ground Control. Tale configurazione dovrebbe essere svolta e accessibile in maniera intuitiva tramite una finestra che permetta di aggiungere e rimuovere i dispositivi e associare loro un determinato setup piuttosto che il software dedicatogli. A livello di codice è presente la classe ConfigurationManager che si occupa di gestire il file di configurazione per quanto concerne i percorsi degli esperimenti, quello predefinito e quello temporaneo, e la lista degli ultimi esperimenti aperti.

### Refactoring

L’applicativo in versione offline funziona, ma a livello di codice vi sono ancora delle parti ambigue che necessiterebbero di analisi al fine di renderle chiare a chiunque possa continuare in seguito a sviluppare il progetto. La risoluzione delle parti ambigue porterà a un refactoring del codice al fine di renderlo più leggibile e renderne più facile la manutenzione.

Il codice sarebbe comunque soggetto a del refactoring, anche se non dovessero emergere delle parti ambigue.

# Conclusioni

Lo svolgimento di questo progetto mi ha permesso di lavorare su vari aspetti sia professionali sia personali.

In primo luogo si è differenziato dagli altri lavori svolti lungo il mio percorso accademico per il suo collegamento concreto con il mondo lavorativo, poiché il prodotto sarà impiegato ed esteso dal team che lavora sui dati gestiti da Ground Control, di cui il relatore prof. Banfi fa parte. Ho potuto constatare come un progetto possa evolversi in corso d’opera e anche come possa essere parzialmente rivisto in base alle esigenze pratiche. Un esempio di questo è stata l’aggiunta dell’immagine di provino degli esperimenti su suggerimento del team di lavoro.

Anche lavorare su dati concreti ha reso l’esperienza più coinvolgente.

Il mio percorso accademico alla SUPSI è stato caratterizzato dall’utilizzo di Java e di JavaFx nel caso delle interfacce grafiche. Questa è stata la mia prima esperienza con il linguaggio C# ed è stata una splendida esperienza! Realizzare l’interfaccia grafica è stato incredibilmente soddisfacente grazie a WPF, poiché ho potuto vedere l’interfaccia prendere forma markup dopo markup in maniera molto responsive. Come citato, le precedenti esperienze concernevano JavaFx dove la programmazione dell’interfaccia grafica, “colpa” anche della mancanza di un editor visuale nell’IDE usata per JavaFx, non mi è mai risultata così comprensibile come su Visual Studio con C# e WPF. Molto probabilmente complice di ciò è il fatto che l’IDE usata possedeva ogni strumento di sviluppo integrato in sé. Ho lavorato con un ambiente di sviluppo, Visual Studio, molto completo per quanto ho potuto appurare con questo progetto.

Per l’esperienza svolta finora, se dovessi sviluppare un’interfaccia grafica la svilupperei in C# con Visual Studio.

Grazie allo studio di Java, l’approccio a C# è stato piuttosto buono dato che entrambi sono linguaggi orientati a oggetti e si ricordano per alcuni aspetti.

Ancora una volta lo svolgimento di un progetto informatico ricorda quanto la community online sia numerosa. La maggior parte delle volte mi sono ritrovato a cercare informazioni nel web con risultati più che buoni, poiché una soluzione è sempre stata trovata sia che fosse ritrovato un codice perfettamente adatto al mio caso, sia che fosse ritrovato uno spunto che portasse poi a un’implementazione scritta da me. E, in aggiunta al punto precedente, quanto anche sia importante sfruttare i controlli e le classi già messe a disposizione dell’ambiente.

Riassumendo: mi sono reso nuovamente conto di quanto sia importante verificare se qualcosa esiste già nella community, se esso è o non è adatto al nostro caso, se ci suggerisce un approccio vincente all’esigenza che abbiamo oppure se si devono cercare delle soluzioni alternative poiché la difficoltà non è superabile.

Un esempio di quanto scritto è stato il controllo WebBrowser messo a disposizione da WPF che è in grado di essere un utilissimo esplora risorse.

A livello personale mi ha ulteriormente fatto maturare, così come a livello professionale, poiché ho potuto impiegare un metodo di lavoro che mi ha permesso di rimanere al passo e di appuntare i problemi riscontrati, le soluzioni trovate, le specifiche implementate, quali scartate o modificate, e via dicendo.

Sono anche piuttosto soddisfatto di come ho gestito il tempo a mia disposizione, anche se l’aggiunta delle varie specifiche mi ha comunque trasmesso la sensazione di essere ugualmente un passo indietro rispetto a quello che avrebbe dovuto essere la mia tabella di marcia.

Le formule che vengono referenziate nel testo vanno identificate con un riferimento numerico progressivo a destra della formula stessa e analogo a quello per le figure e tabelle, nella forma C.N, racchiuso tra parentesi tonde.

Es.: formula: a+b+c = d (2.3),

riferimento: “… come risulta dalla formula (2.3) …”

Piani di Lavoro

Qui va inserita la pianificazione dei lavori di progetto, con almeno il piano iniziale previsto e quello finale di realizzazione effettiva.

Se necessario o utile possono essere inseriti anche dei piani intermedi.

# Fonti

## Bibliografia

Antonio Pelleriti, *Programmare con C#8 | Guida Completa*, Milano, Edizioni LSWR, 2019

## Sitografia

### Pagine

Windows Form

<https://en.wikipedia.org/wiki/Windows_Forms>, ultimo consulto 23 gennaio 2022

Windows Presentation Foundation

<https://en.wikipedia.org/wiki/Windows_Presentation_Foundation>, 23 gennaio 2022

Confronto WPF e WinForms

<https://www.geeksforgeeks.org/difference-between-wpf-and-winforms/>

.NET 6

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

C#

<https://it.wikipedia.org/wiki/C_sharp#Nuove_funzioni_introdotte_con_la_versione_8_del_linguaggio[12>]

Metodo BinarySearch sulle collezioni

<https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1.binarysearch?view=net-6.0>

Ricerca binaria

<https://it.wikipedia.org/wiki/Ricerca_dicotomica>

User control

<https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.usercontrol?view=windowsdesktop-6.0>

### Immagini

.NET 6

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

MVC

<https://upload.wikimedia.org/wikipedia/commons/f/fd/MVC-Process.png>

Appendici

Sono inserite direttamente nel fascicolo della documentazione.

Vanno identificate singolarmente con A1, A2, A3, ecc. e i relativi titoli, oppure a gruppi con A1, A2, ecc., B1, B2, ecc., C1, C2, ecc., nel caso si vogliano evidenziare dei blocchi di appendici dello stesso tipo, come ad es. schemi, diagrammi, listati, ecc.

Nel caso di appendici su cui sia impossibile o troppo complicato inserire la numerazione delle pagine, come ad esempio listati già formattati, va inserita una prima pagina numerata che riporti il contenuto e il numero di pagine dell’appendice stessa, facendo poi seguire le pagine originali (con numerazione autonoma o senza numerazione).

Allegati

Fanno parte della documentazione, ma non del relativo fascicolo, trattandosi di materiale separato, anche se riferito alla documentazione stessa.

Si tratta del CD contenente la documentazione stessa e altro materiale riferito al progetto, ev. fascicoli separati (come ad es. un Manuale d’uso), ev. materiale sperimentale del progetto.

L’identificazione avviene come per le Appendici, usando però lettere iniziali diverse, in modo da non confondere Appendici e Allegati.

Importante: gli Allegati, essendo separati, vanno sempre corredati di scritte o etichette che li identifichino come relativi al progetto e alla documentazione cui si riferiscono (titolo, codice, ecc.).

Nome Cognome (1. Firma) Nome Cognome (2. Firma)

Funzione Funzione

Settore/prodotto Settore/prodotto

1. Capitolo 1 di questo documento [↑](#footnote-ref-1)
2. Antonio Pelleriti, *Programmare con C#8 | Guida Completa*, Milano, Edizioni LSWR, 2019

   , p. 821, rr. 20-21 [↑](#footnote-ref-2)
3. https://it.wikipedia.org/wiki/SOLID [↑](#footnote-ref-3)
4. https://www.winfxitalia.com/articoli/dotnet6/introduzione-dotnet6-csharp10.aspx [↑](#footnote-ref-4)
5. https://it.wikipedia.org/wiki/C\_sharp#Nuove\_funzioni\_introdotte\_con\_la\_versione\_8\_del\_linguaggio[12] [↑](#footnote-ref-5)
6. Antonio Pelleriti, *Programmare con C#8 | Guida Completa*, Milano, Edizioni LSWR, 2019, p. 519, rr. 2-6 [↑](#footnote-ref-6)
7. Informazioni su Visio, controllo 30 gennaio 2022 [↑](#footnote-ref-7)
8. Informazioni su Pencil, controllo 30 gennaio 2022 [↑](#footnote-ref-8)
9. Informazioni su Visual Studio, controllo 30 gennaio 2022 [↑](#footnote-ref-9)
10. <https://it.wikipedia.org/wiki/C_sharp#Nuove_funzioni_introdotte_con_la_versione_8_del_linguaggio[12>, tabella nel capitolo “Storia del linguaggio” [↑](#footnote-ref-10)
11. Informazioni su GitHub Desktop, controllo 30 gennaio 2022 [↑](#footnote-ref-11)
12. https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.usercontrol?view=windowsdesktop-6.0 [↑](#footnote-ref-12)
13. https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1.binarysearch?view=net-6.0 [↑](#footnote-ref-13)
14. Antonio Pelleriti, *Programmare con C#8 | Guida Completa*, Milano, Edizioni LSWR, 2019

    , p. 587 [↑](#footnote-ref-15)