**Documentazione Applicazione Medico**

**Linguaggio di programmazione:** C#

**Ambiente di sviluppo:** Visual Studio 2015

**Modello Dati:** Sql Server

**DBMS:** Sql Server Management Studio 2014

**SINCRONIZZAZIONE**

Ad ogni avvio, l’applicazione effettua una sincronizzazione (tramite chiamata RESTFUL) con il server remoto *(Vedi Fig. 1)*, collegandosi ai servizi API disponibili, così da poter aggiornare i dati contenuti sul DB locale. Al contempo esegue l’invio dei certificati creati successivamente all’ultima data di sincronizzazione.

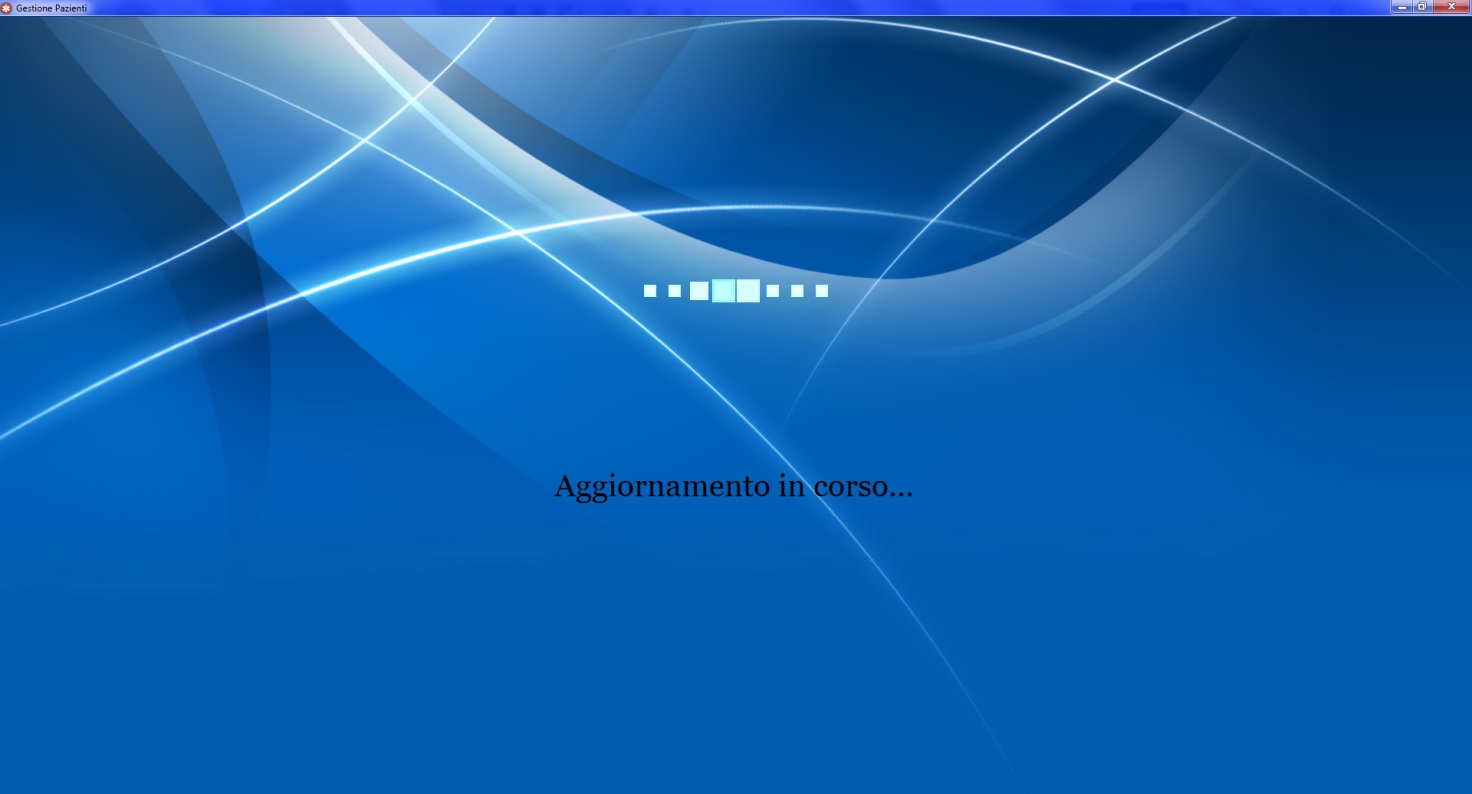
**

Fig. 1

È inoltre possibile effettuare la sincronizzazione in qualsiasi momento scegliendo la voce “Sincronizza” dal menù a discesa della voce File. *(Vedi Fig. 1.1)*

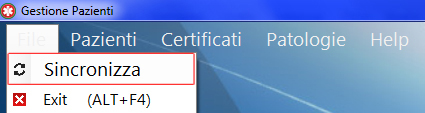


Fig. 2.1

**VISUALIZZAZIONE DATI**

Una volta effettuato l’aggiornamento automatico all’avvio dell’applicazione, l’utilizzatore viene reindirizzato alla sezione di visualizzazione della lista dei pazienti. *(Vedi Fig. 2)*



Fig. 2

Le altre due sezioni (Certificati e Patologie) hanno un aspetto di base similare alla schermata della lista pazienti*.* *(Vedi Fig. 3, 4)*:



Fig. 3

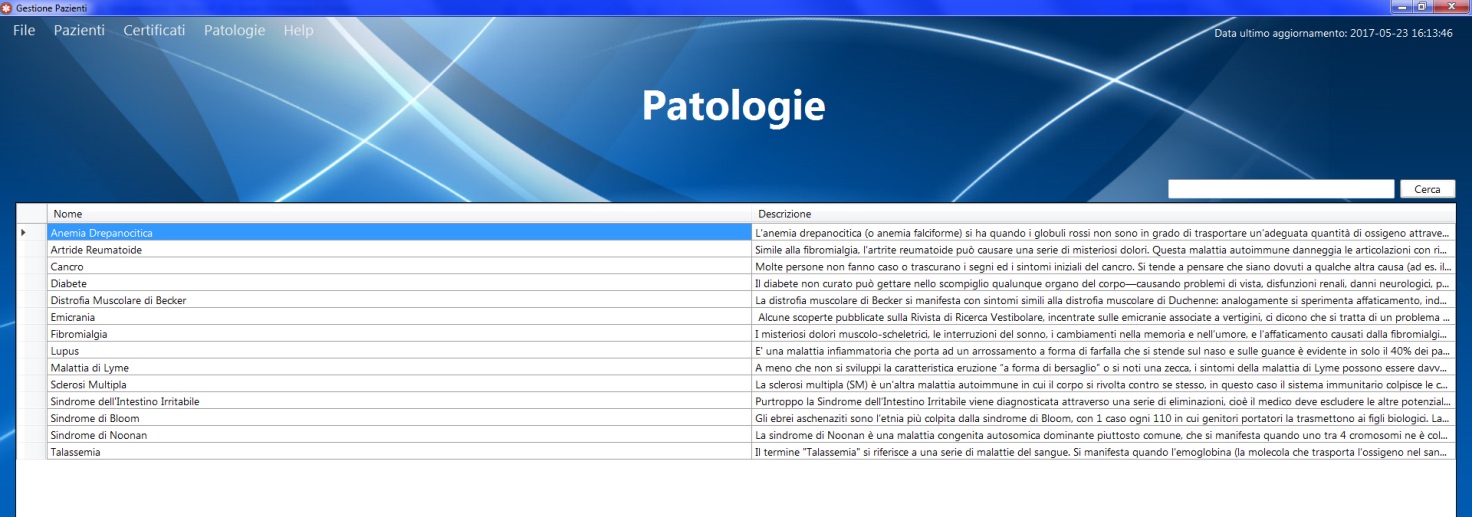


Fig. 4

In ognuna delle tre sezioni precedentemente citate, è possibile filtrare i dati visualizzati attraverso l’apposito campo di ricerca, posizionato in alto a destra della griglia. Tale filtro si applica a tutti i campi della tabella, e avviene dinamicamente alla pressione di un qualsiasi carattere della tastiera, o al click del bottone “Cerca”.

Premendo sull’intestazione di una colonna è inoltre possibile ordinare i dati visualizzati (con carattere ascendente o discendente) in base al testo contenuto all’interno della colonna selezionata.

**SCHEDA DI DETTAGLIO DEL SINGOLO PAZIENTE**

C:\Users\admin\Documents\Alessandro\ProjectWork\Icone\1494256731_info.pngLa griglia dei pazienti si differenzia dalle altre, in quanto presenta all’inizio di ogni riga, l’icona   
 (*Vedi Fig. 2*) che consente di poter accedere alla scheda di dettaglio del paziente desiderato. Tale scheda presenta un’intestazione che varia dinamicamente a seconda del nominativo del paziente selezionato e una griglia contenente i certificati associati a quest’ultimo. *(Vedi Fig. 5)*

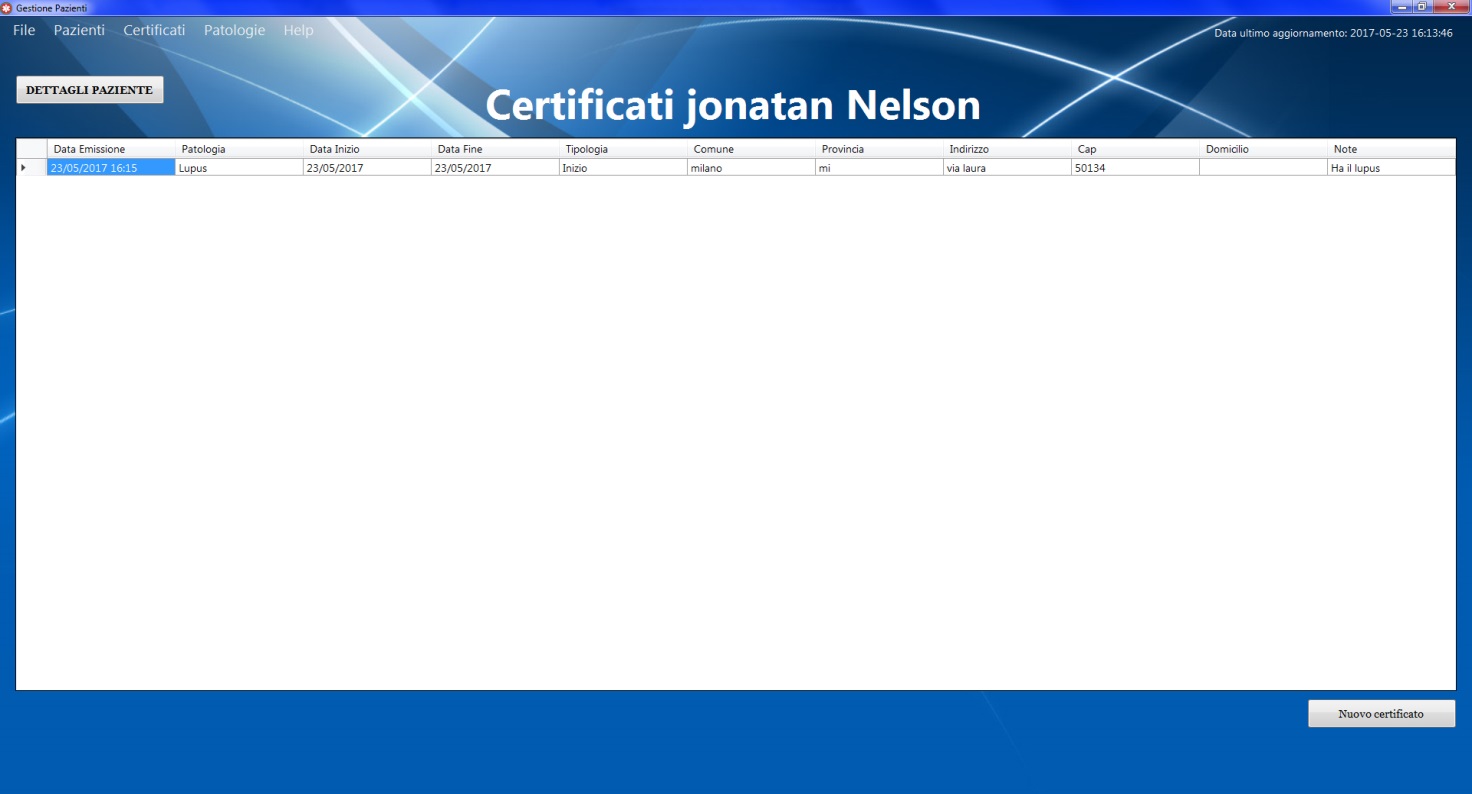


Fig. 5

Inizialmente i dettagli del paziente sono nascosti, ma è sempre possibile visualizzarli facendo click sul bottone “DETTAGLI PAZIENTE”, posto in alto alla sinistra del titolo. *(Vedi Fig. 6)*

**

Fig. 6

Il pulsante “Nuovo certificato” (posto in basso a destra) consente di accedere alla schermata di creazione di un nuovo certificato, partendo da un modello precompilato con i dati del paziente scelto. *(Vedi Fig. 7)*

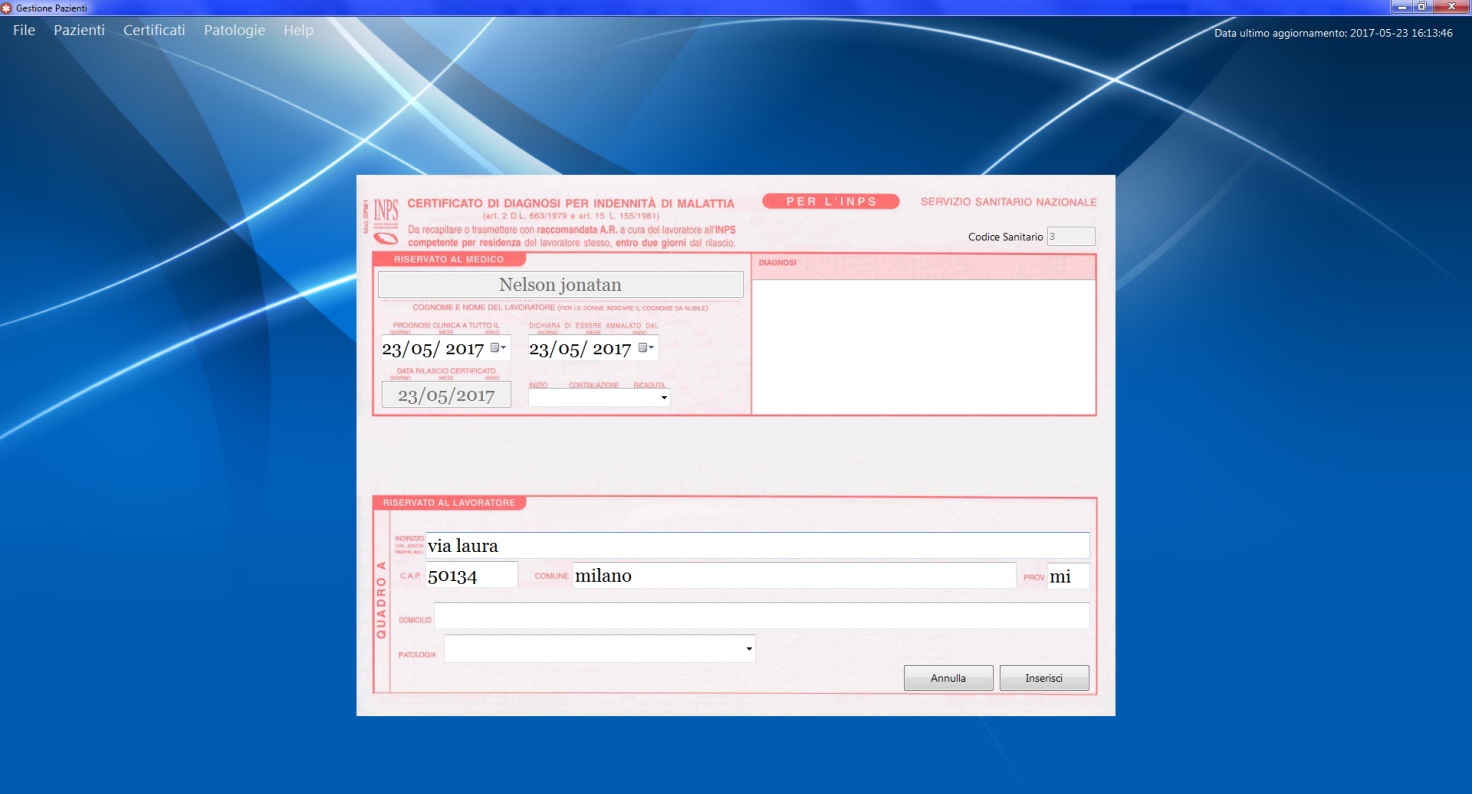
**

Fig. 7

Premendo sul pulsante **Inserisci** il certificato verrà aggiunto sul database come nuovo record e di conseguenza visualizzato nella griglia generale, e anche in quella del singolo paziente.  
L’invio al server remoto, del certificato appena inserito, avverrà al riavvio dell’applicazione o al click della voce di menù **Sincronizza**, come illustrato in Fig. 1.1  
Premendo invece sul pulsante “Annulla” l’utilizzatore verrà reindirizzato alla schermata precedente, ovvero alla scheda di dettaglio del paziente.

**ALTRE FUNZIONALITA’**

La voce **Exit** presente nel menù a discesa della voce **File** permette di chiudere l’applicazione. (E’ possibile anche farlo con lo shortcut **ALT+F4**)

La voce di menù **Help** mostra i nominativi degli sviluppatori del presente software.

In ogni momento è possibile visualizzare data e ora dell’**ultima sincronizzazione** col server, poste nell’angolo in alto a destra della finestra.

**DOCUMENTAZIONE TECNICA**

Il “cuore” dell’applicazione risiede principalmente in 3 funzionalità:

1. Ricezione di dati nuovi e/o aggiornati dal server remoto
2. Invio dei certificati creati dopo l’ultima data di sincronizzazione
3. Serializzazione e de-serializzazione delle informazioni da inviare/ricevere

Come si può vedere in *Fig. 8*, per prima cosa l’applicazione crea una nuova istanza della classe **WebClient**.   
WebClient è una classe presente nella libreria .NET che consente di comunicare con i servizi API esposti da un server, tramite protocollo RESTFUL.  
Nel nostro caso viene utilizzato sia per la ricezione che per l’invio dei dati.

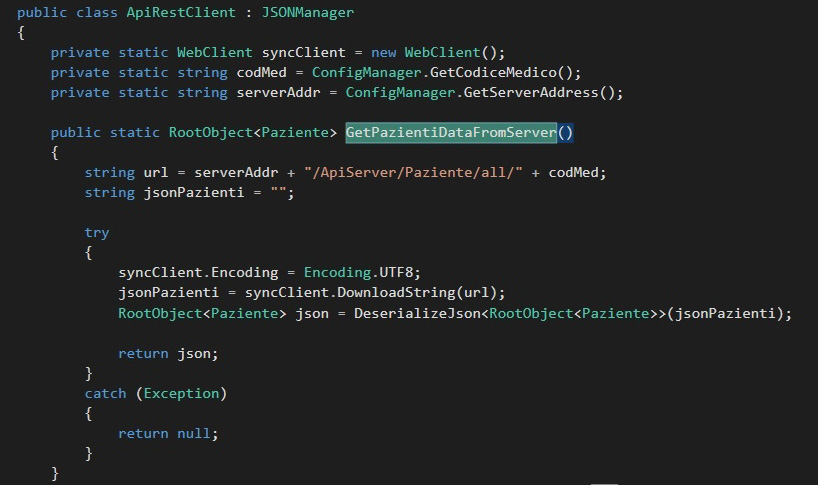


Fig. 8

Subito dopo, tramite due metodi statici presenti all’interno della classe **ConfigurationManager** (creata dagli sviluppatori), si vanno a leggere da un file di configurazione, inserito per semplicità nella cartella dell’applicazione, l’indirizzo IP del server remoto e il codice del medico che sta accedendo all’applicazione. *(Vedi Fig. 9)*  
In un’applicazione più strutturata ovviamente, tali dati verrebbero inseriti in una tabella del DB e di conseguenza verrebbe creato un meccanismo di login all’avvio dell’applicazione per consentire l’utilizzo del medesimo software anche a più medici.

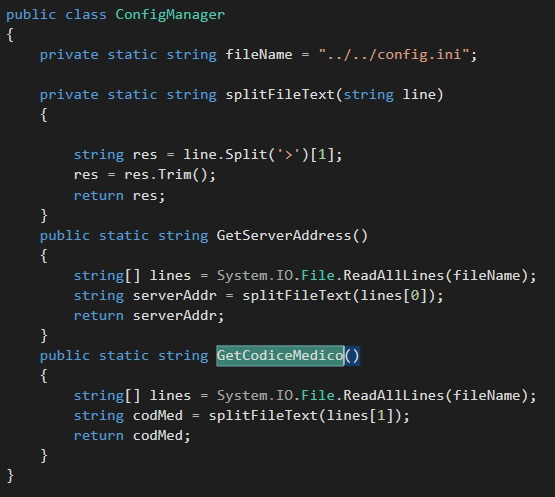


Fig. 9

Tornando alla *Fig. 8* si può vedere che il metodo **GetPazientiDataFromServer()** crea una stringa contenente la concatenazione dell’indirizzo IP del server remoto, del servizio API esposto e del codice medico, così da poterla passare come parametro al metodo **DownloadString** dell’istanza di classe di **WebClient**. Tale metodo restituisce una stringa (in formato JSON) contenente tutti i pazienti associati ad un singolo medico. Tali dati per essere processati correttamente dall’applicazione devono essere de-serializzati e ciò avviene tramite il metodo **DeserializeJson** della classe **JsonManager** *(vedi Fig. 10).*

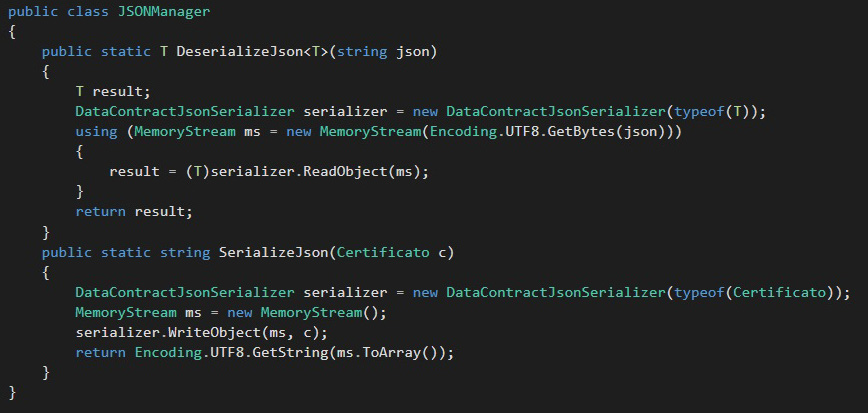
**

Fig. 10

Entrambi i metodi esposti dalla classe **JsonManager** utilizzano la classe **DataContractJsonSerializer** e la classe **MemoryStream** per poter trasformare i dati ricevuti (o quelli da inviare) in un formato comprensibile dall’applicazione (o dal server remoto).  
Nel primo caso il metodo riceve una stringa in ingresso (ovvero il JSON risultante dalla chiamata REST) e restituisce un oggetto generico, così da non dover implementare un metodo per ogni classe di oggetti presente nel programma.  
Il secondo metodo invece riceve come parametro di ingresso un oggetto della classe Certificato, lo serializza, e restituisce una stringa formattata ad hoc per l’invio al servizio API relativo.  
Questo secondo metodo viene richiamato all’interno di un altro metodo dell’applicazione   
*(Vedi Fig. 11)* il quale, eseguendo un ciclo **foreach** su una lista di certificati, richiama il metodo **UploadString** della classe **WebClient**, passando come parametri in ingresso l’indirizzo del servizio API e la stringa JSON serializzata, così da poter inviare al server remoto ogni certificato non ancora spedito.

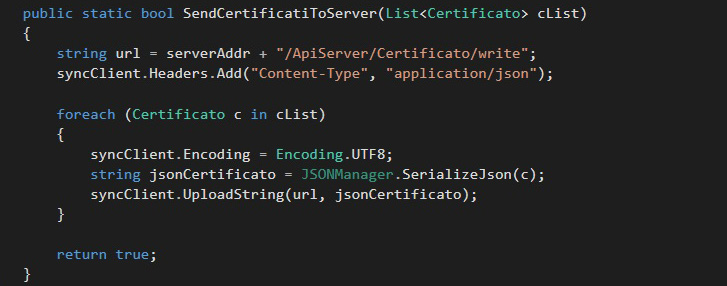


Fig. 11

Il resto dell’applicazione è stato costruito a partire da oggetti già presenti all’interno dell’ambiente di sviluppo.  
Ad ogni oggetto sono stati associati uno o più eventi, i quali una volta “scatenati” eseguono specifiche azioni impostate dagli sviluppatori, così da dare un comportamento ben preciso ad ogni elemento presente sull’applicazione.

La grafica è stata creata interamente dagli sviluppatori. Le parti più significative sono:

* Ogni sezione è un pannello che viene mostrato e nascosto all’esigenza
* Il **loader** iniziale, è un’immagine GIF in loop, alla quale sono stati applicati 5 secondi di **stallo** (gestito da un **Timer**) per dare l’impressione di un vero e proprio caricamento.
* L’immagine del **nuovo certificato**, la quale prende spunto da un vero e proprio certificato medico lavorativo, ma è stata editata per poter “ospitare” le caselle di testo contenenti date e informazioni del paziente.

Inoltre sono stati implementati due metodi di “aiuto” che consentono ad ogni oggetto grafico presente sull’applicazione di avere una posizione e una dimensione definita programmaticamente, così da ottenere un risultato visivo ottimale per l’utilizzatore.