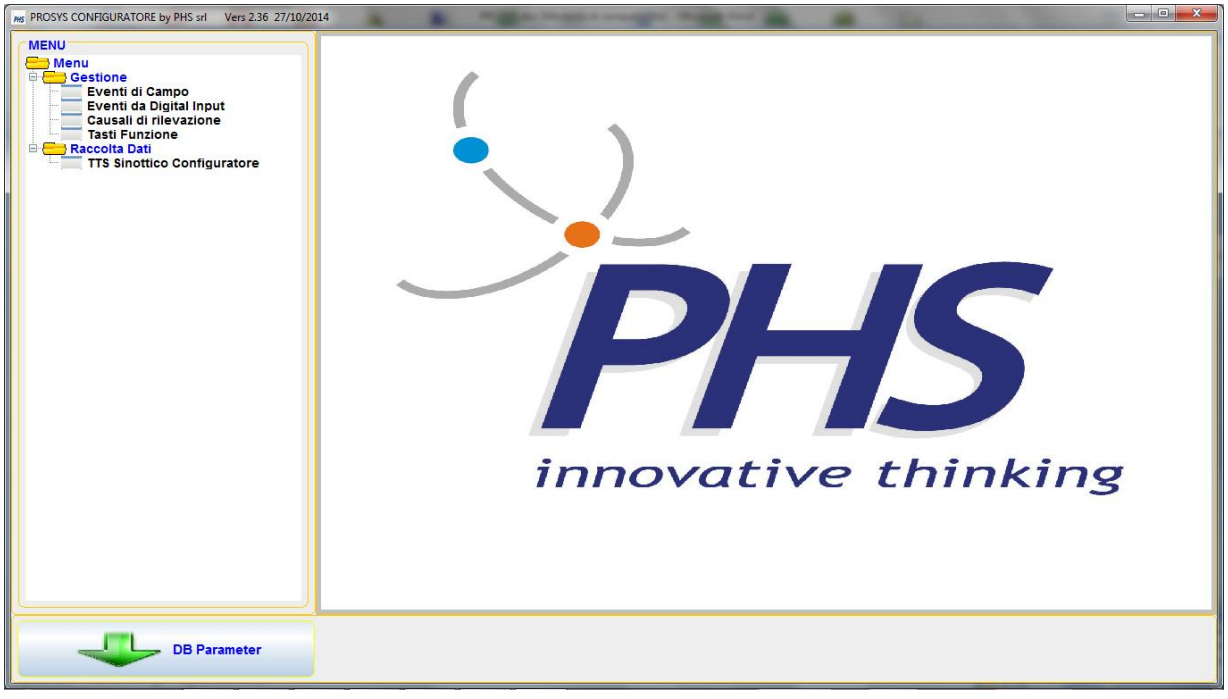
®

PHS net
always innovative

PROSYS



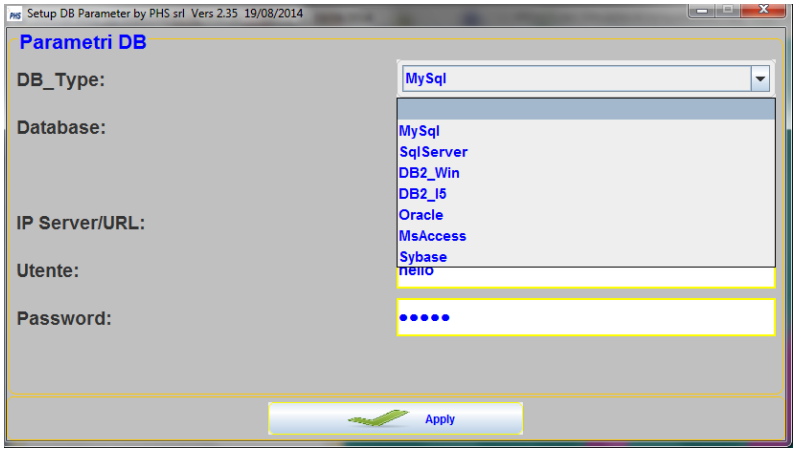
04.02.2015
Vers. 1.1

Indice

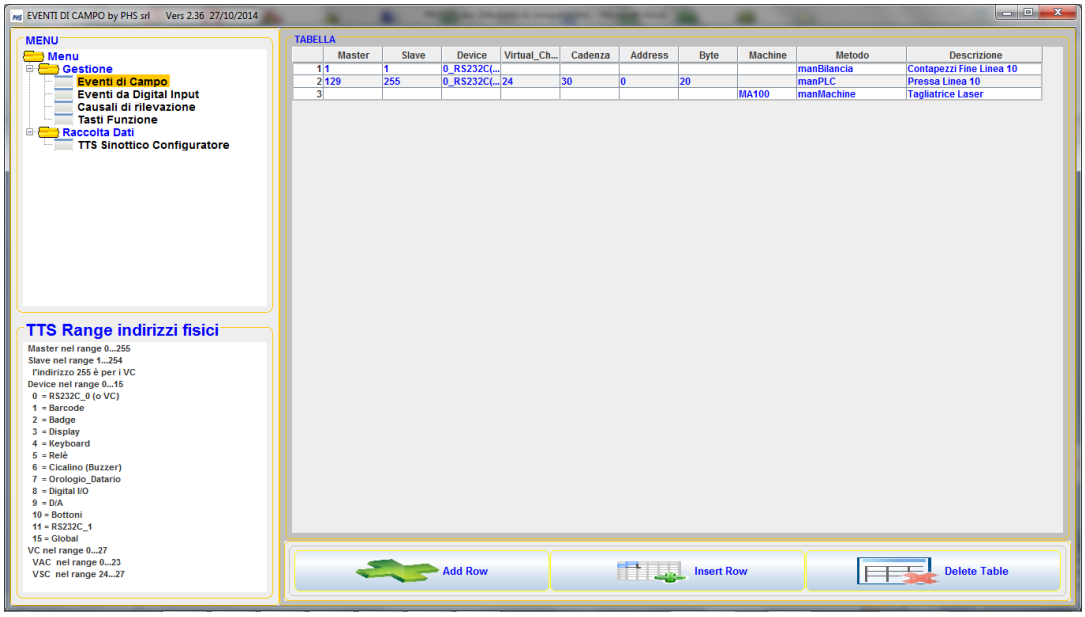
⇐ INDICE

Generalità

Il Prosys è un'applicazione, basata sul framework TTSMAN, che permette di gestire delle tabelle su un Database tra quelli implementati.



Le tabelle permettono di associare a **dei criteri** (eventi di campo, causali,ecc.) **dei metodi**.



I metodi sono richiamati quando il Prosys riconosce il criterio impostato.

I metodi devono appartenere ad un Plugin sviluppabile in Java o .NET.

Il Prosys è composto da due applicazioni fornite tramite due file .jar:

- la prima (**KeepProsys**), priva di form, è associabile ad un servizio Windows o avviata allo Startup, e gestisce gli eventi dichiarati nelle tabelle richiamando i metodi relativi inseriti nel Plugin;

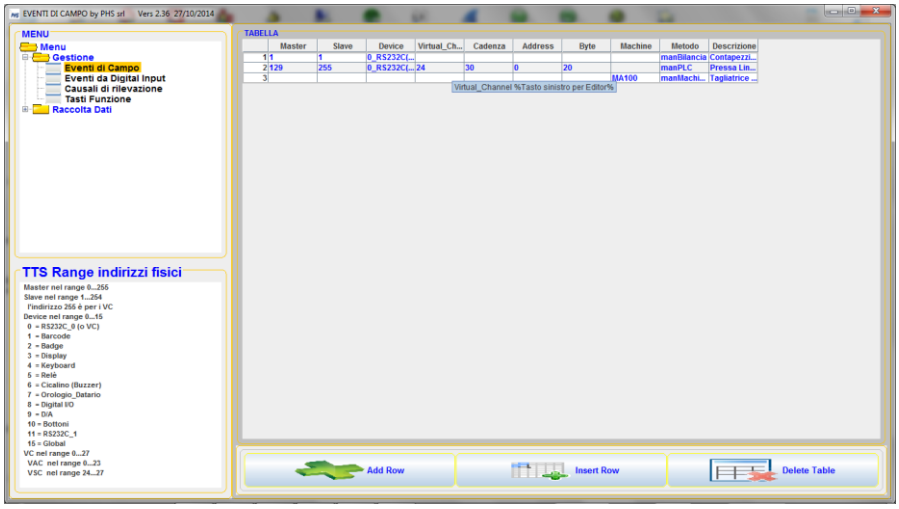
- la seconda (**ConfProsys**) permette di compilare tutte le tabelle relative ai criteri scelti ed alle configurazioni degli oggetti del sistema di raccolta dati TTS.

⇐ INDICE

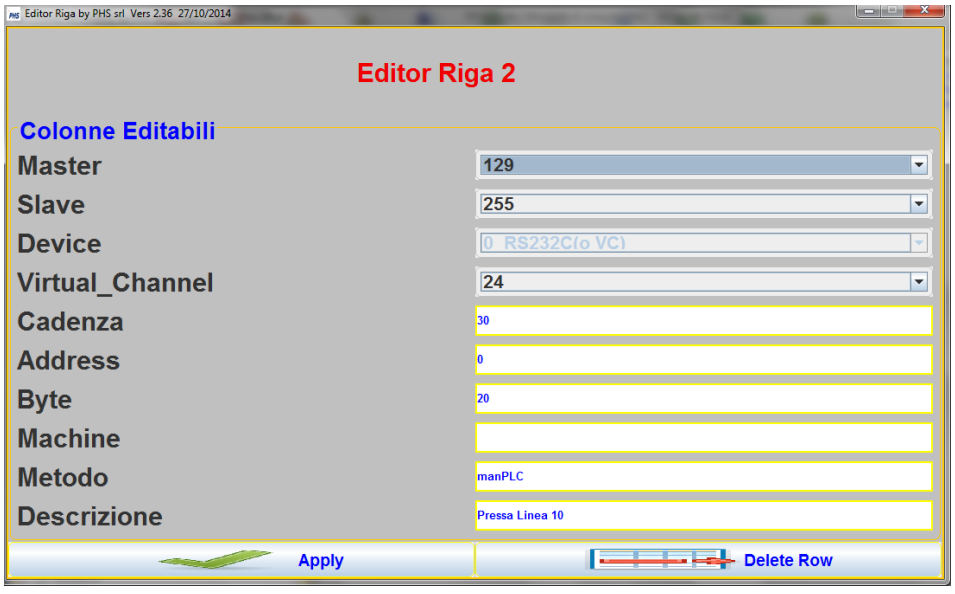
Editor delle Tabelle

Sul pannello **Prosys Configuratore,** il bottone **DB Parameter** permette di configurare in Data Base voluto.

Per le altre tabelle le modalità di editor sono:



- **Editor Riga**: il tasto sinistro su una riga (come indicato dal tooltip) attiva un form del tipo seguente:



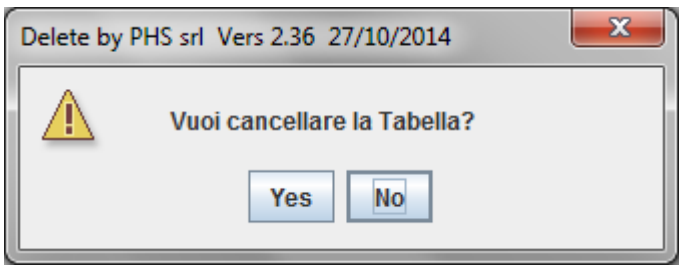
con questo è possibile modificare i campi e confermarli (**Apply**) o cancellare la riga (**Delete Row**).

⇐ INDICE

- **Add Row**: con il bottone relativo si attiva il pannello seguente, che permette di aggiungere una riga in fondo alla tabella.

- **Insert Row**: con il bottone relativo si attiva il pannello seguente, che permette di inserire una riga all'interno della tabella a partire dalla riga dichiarata.

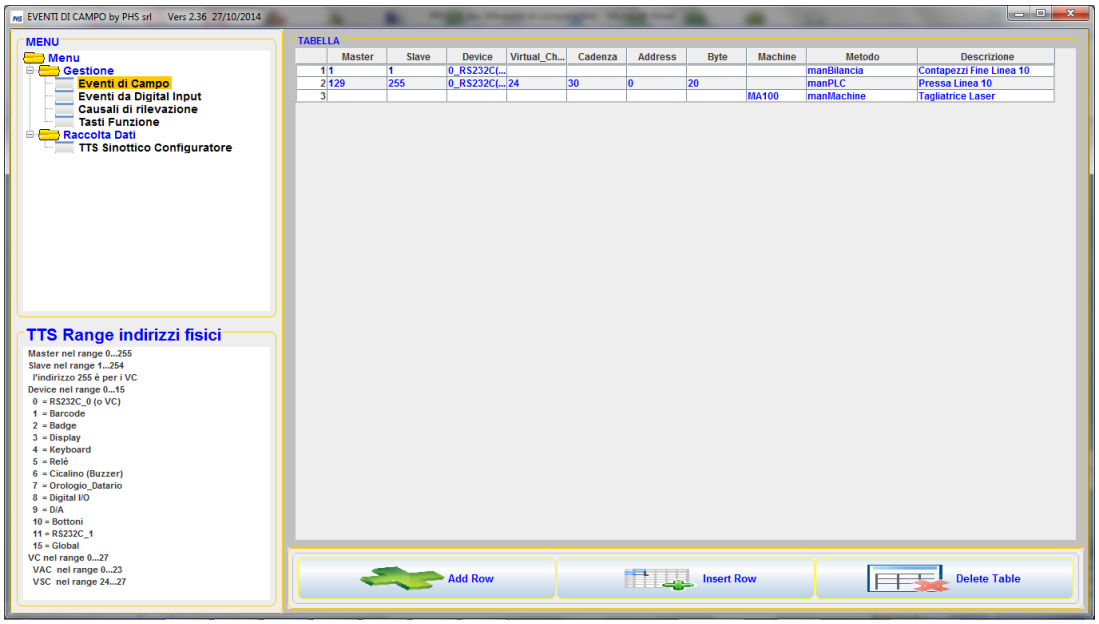
- **Delete Table**: con il bottone relativo è possibile cancellare la tabella, in questo caso viene chiesta la conferma.



⇐ INDICE

Eventi di campo

La tabella “Eventi di campo” è relativa ai dispositivi di comunicazione o di data entry.



I dispositivi di comunicazione sono:

- RS232C_0 (porta seriale);
- RS232C_1 (porta seriale);
- VAC (canali virtuali asincroni UDP/IP);
- VSC (canali virtuali sincroni TCP/IP)
- MACHINE (dispositivi non TTS dotati di protocollo di comunicazione TCP/IP o UDP/IP).

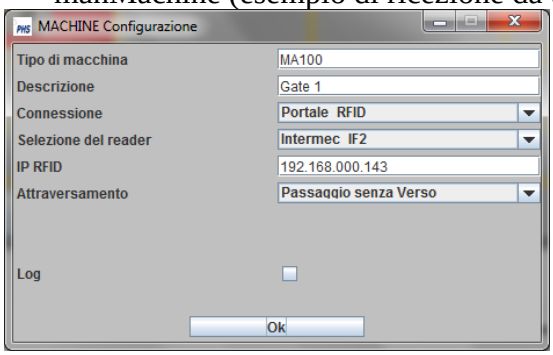
I dispositivi di data entry sono:

- BARCODE;
- BADGE;
- KEYBOARD;

ai dispositivi di data entry può essere associato un metodo, in questo caso il dispositivo associato è sottratto alla gestione delle Causali di rilevazione descritte più avanti.

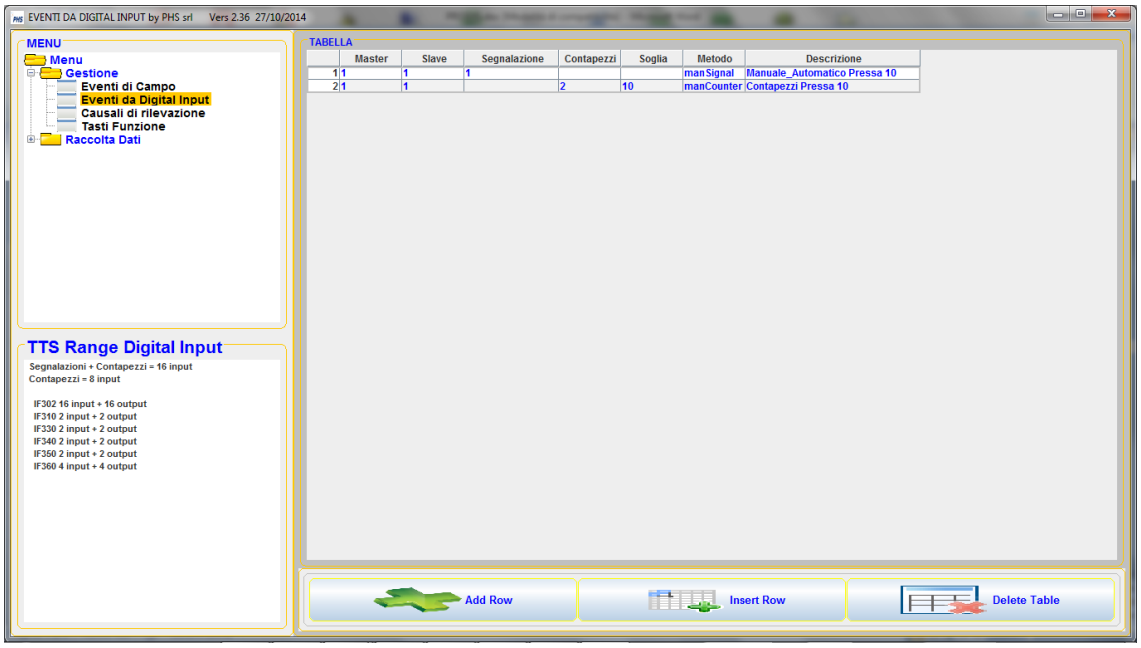
Nel plugin di esempio è riportata la struttura per i metodi:

- manBilacia (esempio di ricezione dalla porta seriale);
- manPLC (esempio di ricezione da un canale VSC relativo ad un PLC Siemens S7);
- manMachine (esempio di ricezione da un Portale RFID con reader Intermec IF2).



⇐ INDICE

Eventi da Digital Input



Nel sistema TTS esiste la porta digital I/O, gli input sono configurabili in due modalità:

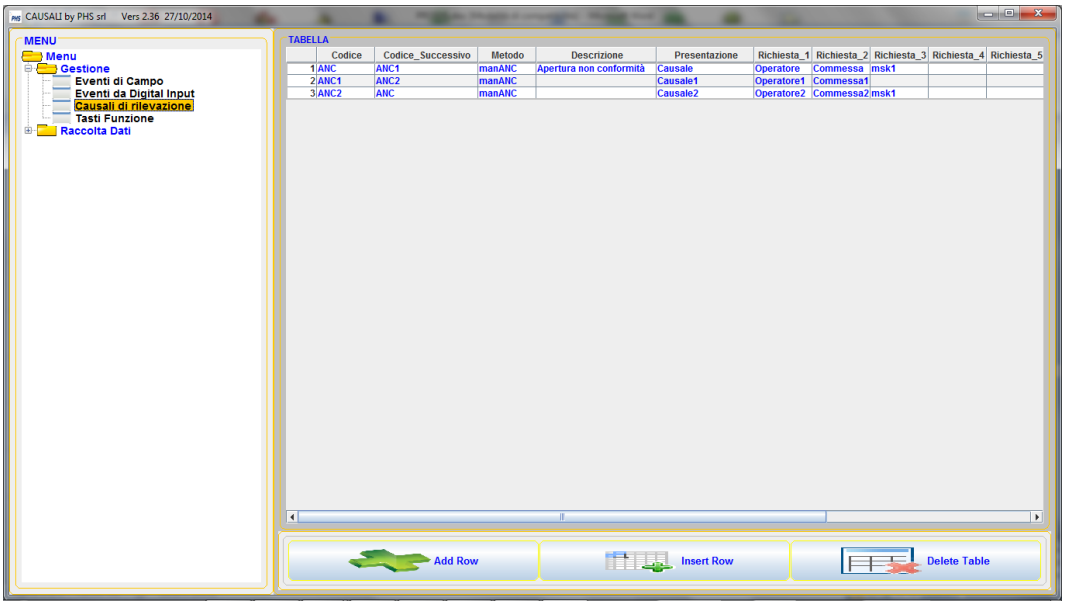
- **Segnalazione**: in questo caso vengono trasmesse le transizioni off-on e on-off relative all'input;
- **Contapezzi**: viene trasmesso il conteggio delle transizioni off-on, secondo i parametri impostati (soglia, cadenza ecc.).

Nel plugin di esempio è riportata la struttura per i metodi:

- manSignal (esempio di ricezione di segnalazione, stato della macchina, allarmi, ecc.);
- manCounter (esempio di ricezione di un contapezzi).

⇐ INDICE

Causali di rilevazione



La tabella “Causali di rilevazione” regola il data entry definendo una sequenza di richieste ll’operatore; i campi sono:

- **Codice**: definisce il codice che bisogna digitare o leggere (barcode o badge) per aprire la causale;
- **Codice_Successivo**: definisce l’eventuale sequenza successiva da attivare al termine della prima;
- **Presentazione**: rappresenta la videata presente sul display in attesa dell’inserimento del codice della causale;
- **Richiesta_1 …. Richiesta_10**: identificano le richieste da presentare all’operatore.

L’uso del Codice_Successivo permette di creare delle catene di richieste come dei loop; la sequenza definita nell’esempio è:

- Causale;
- Operatore;
- Commessa;
- msk1 (display formattato);
- Operatore1;
- Commessa1;
- Operatore2;
- Commessa2;
- msk1;
- Operatore;
- ecc.

si crea, quindi, una sequenza in loop.

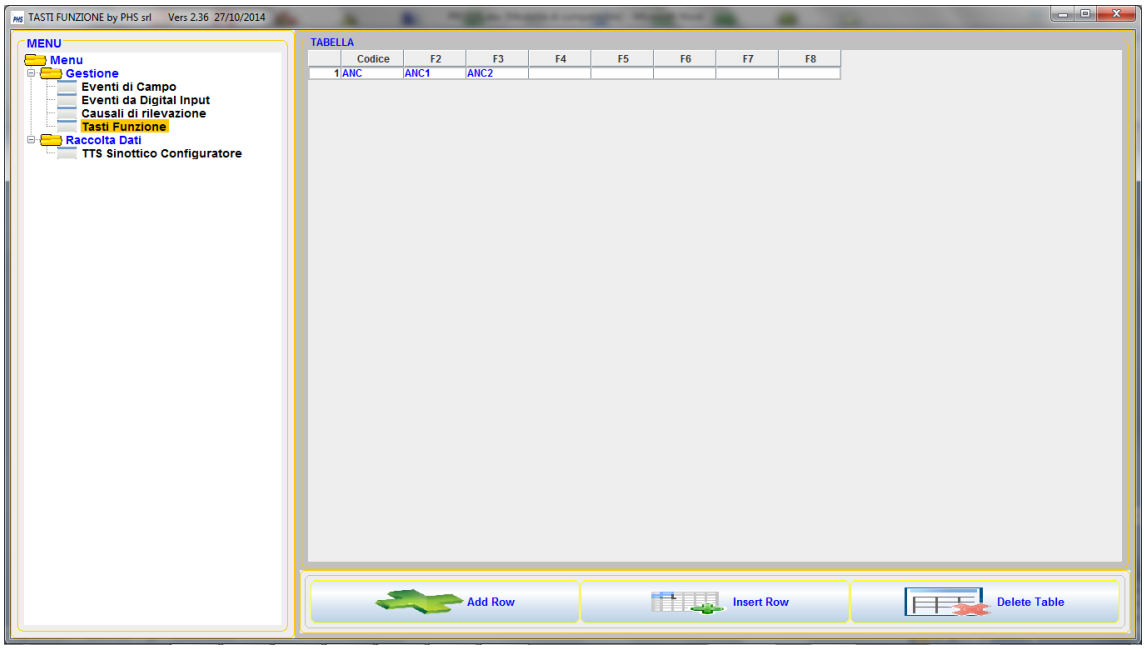
I dispositivi che sono usati per la rilevazione sono: BARCODE, BADGE e KEYBOARD, se non utilizzati per gestire Eventi di Campo.

Il tasto funzione F1 è dedicato a resettare la sequenza di richiesta riportando alla fase di Presentazione.

Per tutte le transazioni generate dal data entry il Prosys richiama il metodo dichiarato in tabella, nell’esempio manANC, con il quale si ne ha controllo.

⇐ INDICE

Tasti Funzione

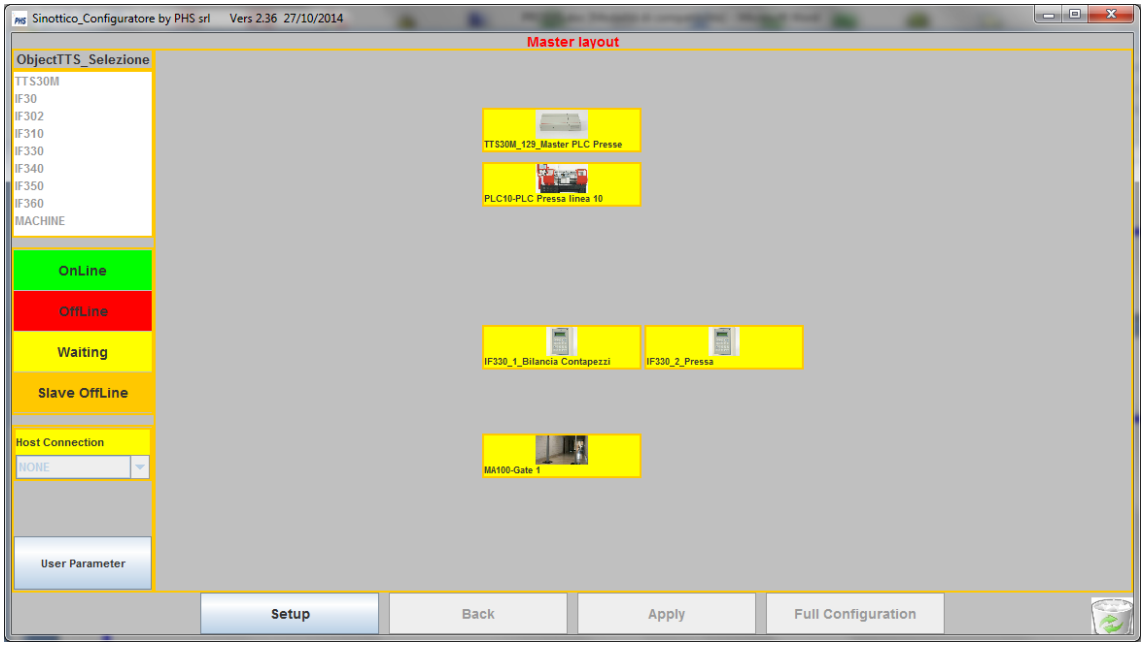


La tabella “Tasti Funzione” associa ad una causale i tasti funzione, con i quali si può cambiare la sequenza; nell’esempio, quando è in corso la sequenza relativa alla causale ANC,:

- F2 attiva la sequenza legata alla causale ANC1;
- F3 attiva la sequenza relativa alla causale ANC2.

⇐ INDICE

Sinottico_Configuratore



Il Sinottico_Configuratore (descritto nel manuale TTSSDK) prevede due tipi di oggetti:

- **oggetti TTS**: sono tutti gli apparti hardware del TTS (TTS30M, IF30, ecc.);
- **oggetti MACHINE**: sono tutti gli apparati hardware dotati di un protocollo TCP/IP o UDP/IP su porta ethernet non prodotti dalla PHS.

Nella modalità “Sinottico” viene indicato, con il colore opportuno, lo stato della connessione degli oggetti configurati.

Nella modalità “Configuratore” si possono aggiungere e configurare gli oggetti secondo la struttura dell’impianto disegnato.

⇐ INDICE

Plugin

Il Plugin permette di sviluppare i metodi, previsti nelle tabelle, per elaborare le transazioni generate dai dispositivi di campo.

Il plugin è sviluppabile in due versioni:

- **plugDataProsys.jar**: versione Java;
- **UserProsys.netmodule**: versione .NET.

La versione Java, se presente nella cartella principale, è prevalente ed in sua assenza è caricata la versione .Net, che deve essere accompagnata dalla dll **JniProsys.dll**.

L'applicazione .Net deve essere **compilata** (non può essere una dll) e della quale vengono forniti i .bat sia per C# che VB.

La struttura di un metodo configurabile in tabella è:

public ArrayList **method**(Object[] parmObj,String[] CmaskF)

dove:

- **parmObj** è un vettore dei parametri che vengono passati;
- **CmaskF** è un vettore di stringhe relative alle possibili videate di un display TTS, che è null quando non è significativo;

sono forniti gli esempi dei metodi descritti in precedenza per le tabelle del Prosys.

La versione Java utilizza il framework **TTSman_fat.jar**, disponendo così di tutte le sue classi (gestione dei Data Base, dei socket, ecc.); per la versione .NET esiste la classe **FunJava**, che contiene dei metodi che richiamano delle funzioni significative dell'ambiente Java ed altre saranno fornite a seconda delle esigenze; inoltre, è implementata la gestione dei Data Base.

Esistono dei metodi cablati, che permettono delle azioni significative per il sistema TTS, i quali hanno delle differenze per le due piattaforme Java e .NET; di seguito sono elencati evidenziando la differenza se esiste.

Il metodo **inizialize** è richiamato alla partenza delle applicazioni Prosys e permette di instanziare gli oggetti necessari per la struttura del plugin:

public void **inizialize**(Object[] parmObj) versione Java;

public void **inizialize**(Object eCsInvoke) versione .NET;

eCsInvoke è un oggetto che passa la dll JniProsys, che serve a richiamare le funzioni dell'ambiente Java.

Per i tasti funzione, se non esiste un'altra gestione, è richiamato il metodo:

public ArrayList **manFunction**(Object[] parmObj,String[] CmaskF).

⇐ INDICE

Per i bottoni di un IF360 è richiamato il metodo:

public ArrayList **manButtons**(Object[] parmObj,String[] CmaskF).

Per un oggetto di tipo "MACHINE" è richiamato il metodo:

public ArrayList **initMachine**(Object[] parmObj,String[] CmaskF);

per l'inizializzazione della connessione ed il metodo:

public ArrayList **parceMachine**(Object[] parmObj,String[] CmaskF);

per un evento generato da una macchina di tipo "Machine", per permettere il parsing o il filtro di un campo (ad esempio per assemblare i record di un file generico in presenza di un FTP).

Per un canale VAC è richiamato il metodo:

public ArrayList **rcvHostVAC**(Object[] parmObj,String[] CmaskF);

il metodo permette di organizzare la stringa da inviare ad un dispositivo collegato ad un canale VAC, quando la transazione parte dall' host su cui è caricata l'applicazione di raccolta dati.

Quando termina la configurazione di un dispositivo TTS, è richiamato il metodo:

public ArrayList **startUP**(Object[] parmObj,String[] CmaskF).

Quando termina lo scarico del buffer di un dispositivo TTS nello stato di batch, è richiamato il metodo:

public ArrayList **manEndBuffer**(Object[] parmObj,String[] CmaskF).

Solo per l'applicazione **ConfProsys,** esistono i metodi:

- **displayParm** richiamato dal bottone "**Altri Dati"** di un oggetto TTS:

public ArrayList **displayParm**(Object[] parmObj) versione Java;

public ArrayList **displayParm**(Object[] parmObj,String[] dummy) versione .NET.

- **displayMach** richiamato dal bottone "**Altri Dati"** di un oggetto Machine:

public ArrayList **displayMach**(Object[] parmObj) versione Java;

public ArrayList **displayMach**(Object[] parmObj,String[] dummy) versione .NET.

- **displayUser** richiamato dal bottone "**User Parameter"**:

public ArrayList **displayUser** (Object[] parmObj) versione Java;

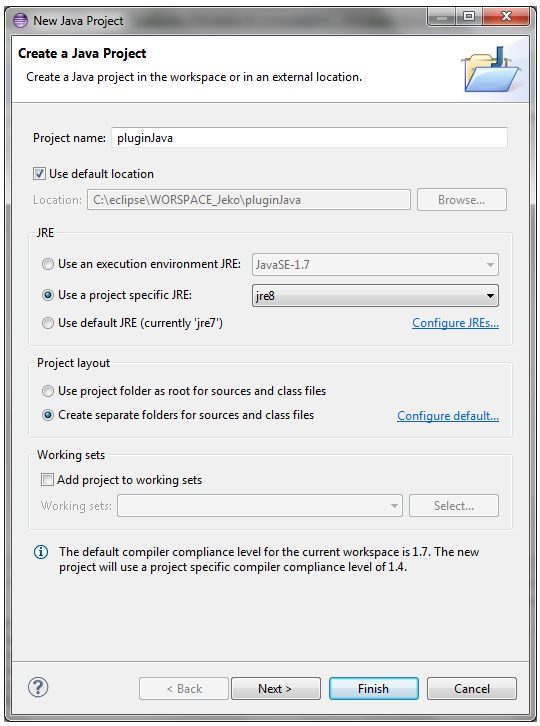
public ArrayList **displayUser** (Object[] parmObj,String[] dummy) versione .NET.

⇐ INDICE

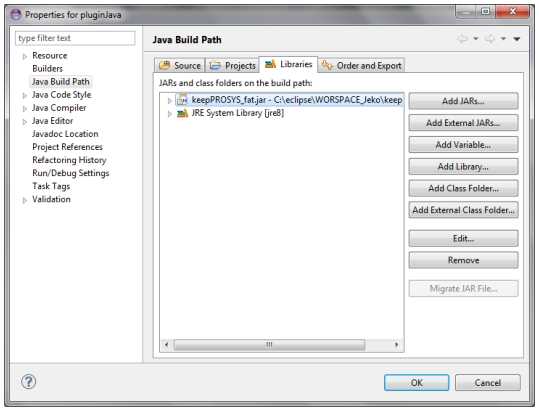
Sviluppo del plugin in Java

Per lo sviluppo del plugin, installato il Prosys, i passi sono:

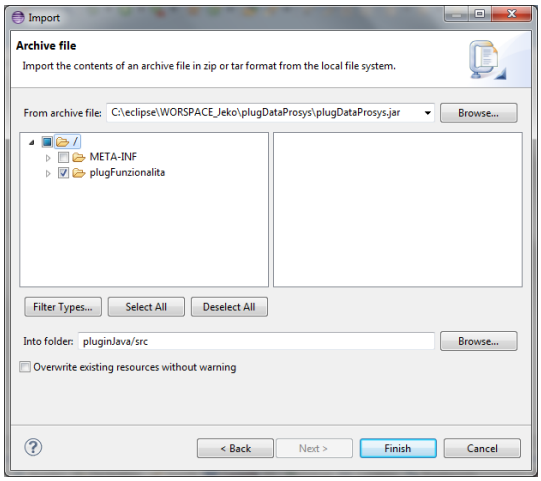
- creazione di un **progetto Jav**a (es.pluginJava);



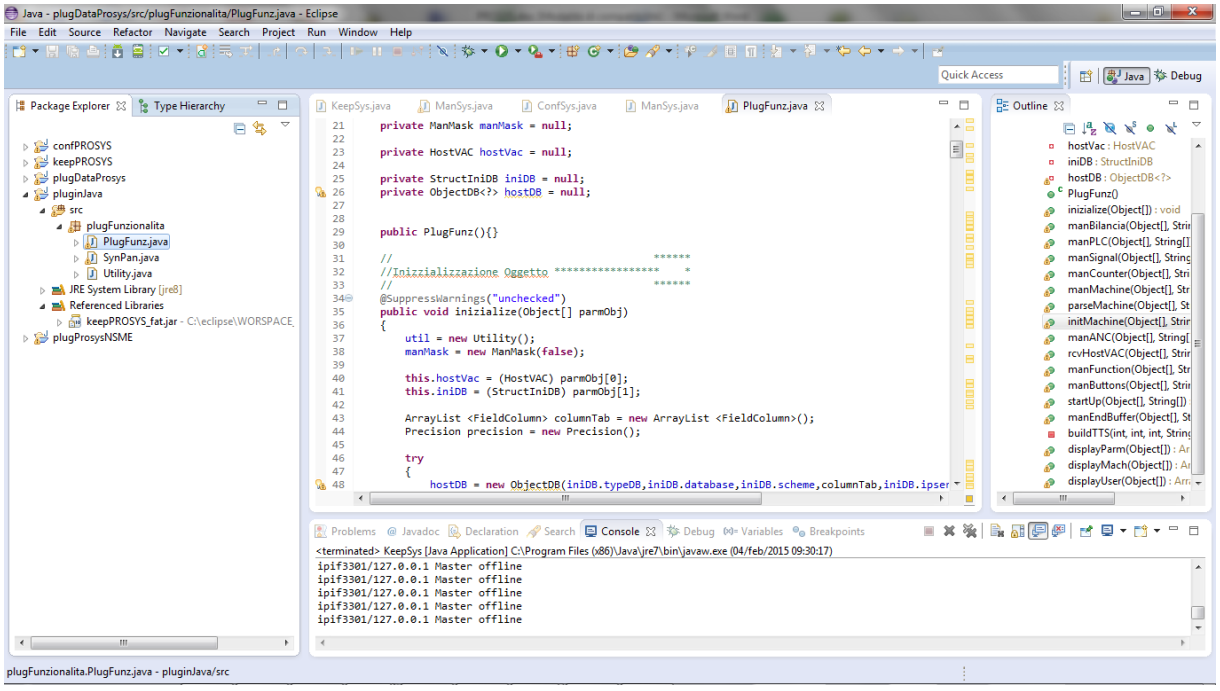
- aggiunta del file .jar **keepProsys_fat**;



- importazione, nel **package src,** del **plugDataProsys.jar generico**, che contiene i sorgenti e le classi predefinite.

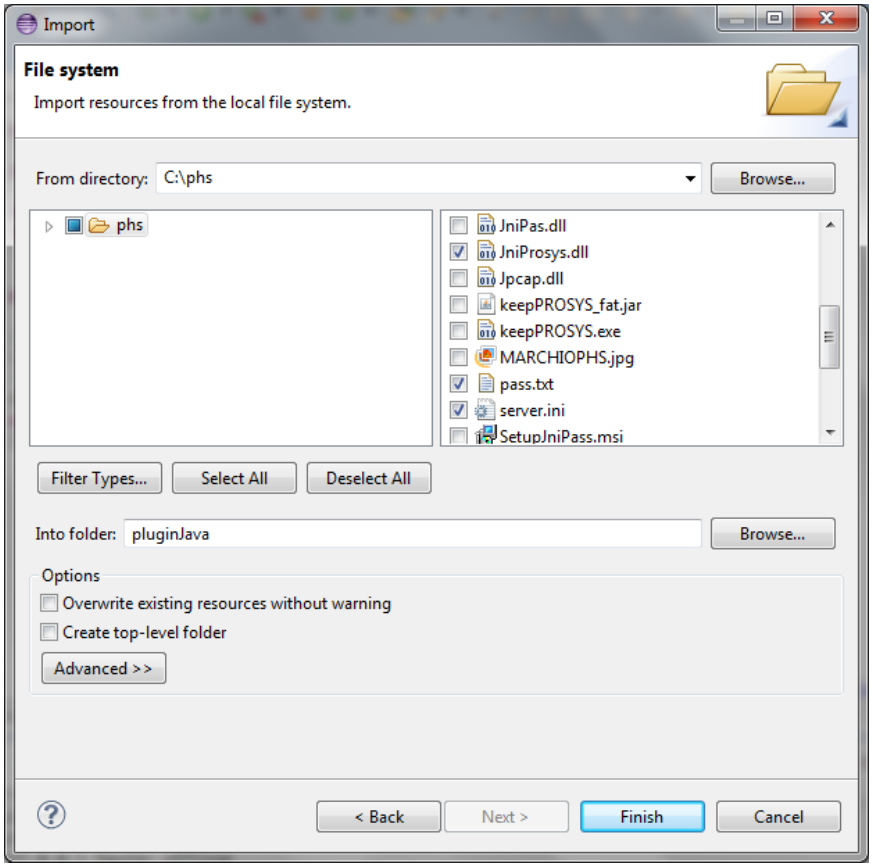


Si ottiene una struttura del tipo:

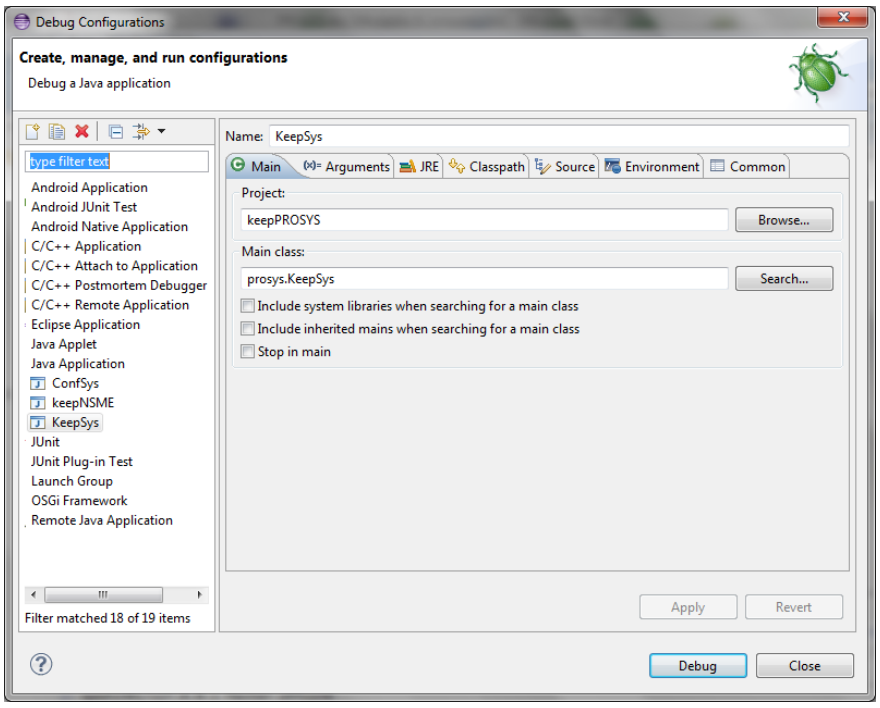


⇐ INDICE

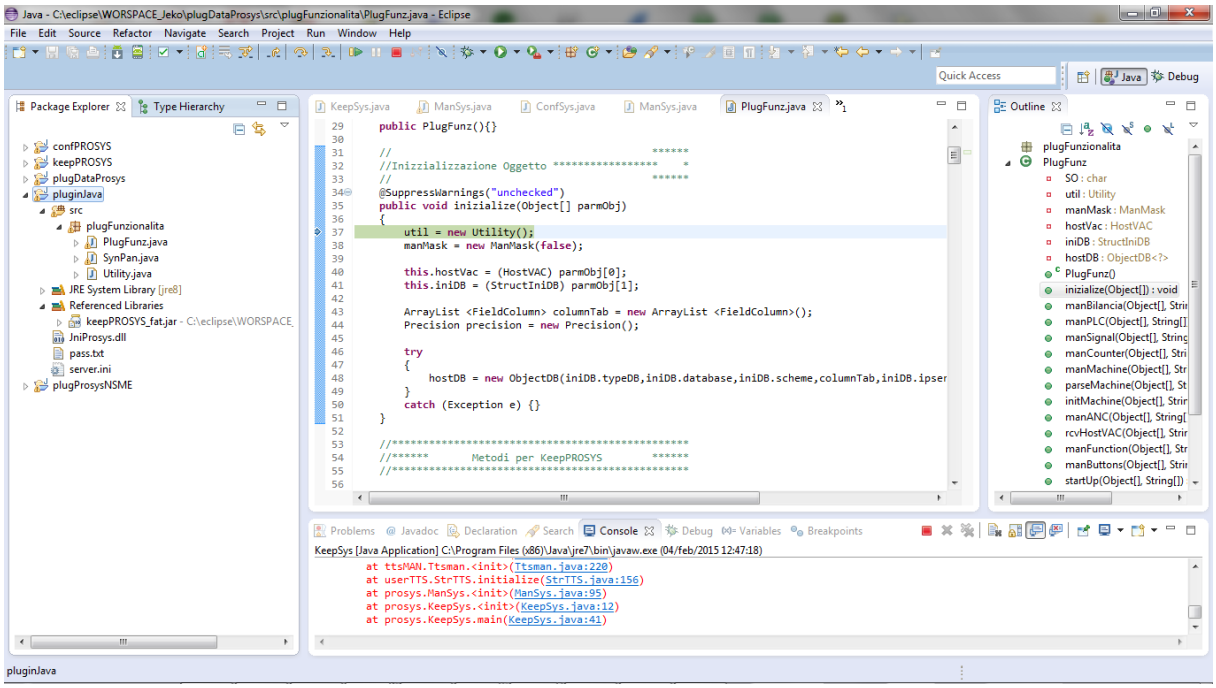
- importazione, come **File System,** della cartella contenente l'installazione (es. PHS) ;



- selezione del **main** nel package **prosys,** ed avvio del debug;



si ottiene:



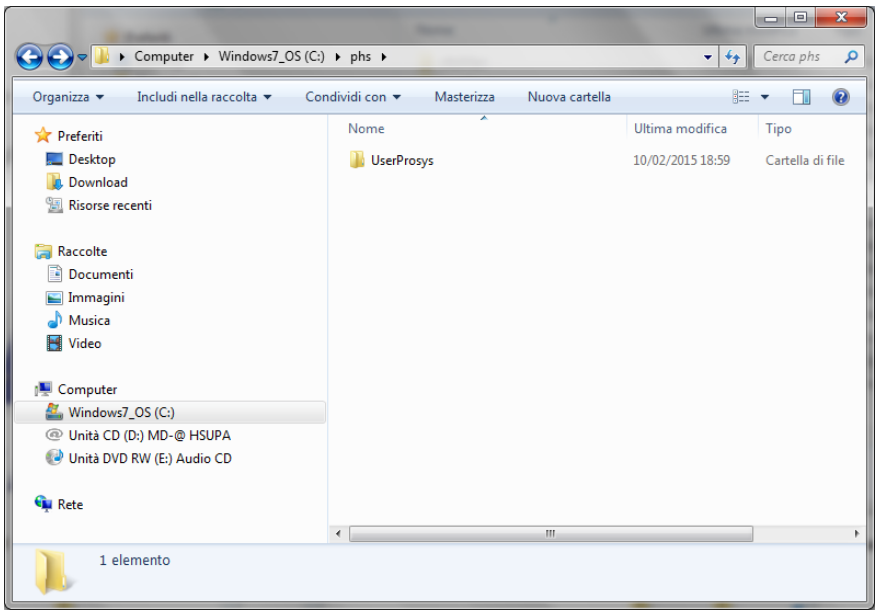
I metodi della classe **PlugFunz** vanno sviluppati secondo le proprie esigenze; terminato lo sviluppo, bisogna esportare il plugin, come **plugDataProsys.jar**, nella cartella contenente l'installazione del Prosys (es. PHS).

⇐ INDICE

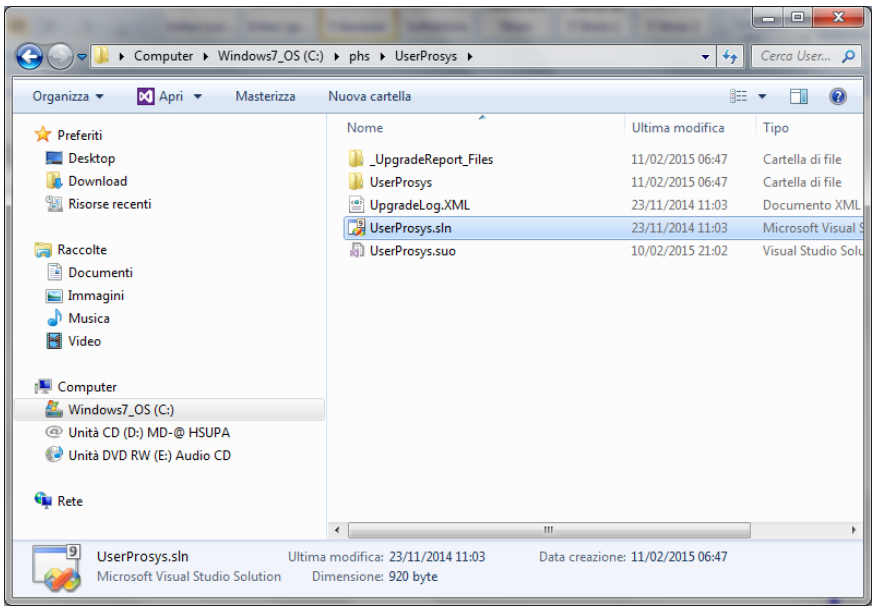
Sviluppo del plugin in .NET

Per lo sviluppo del plugin, installato il Prosys, i passi sono:

- creazione di un progetto in Visual Studio, partendo dal progetto UserProsy fornito come esempio;

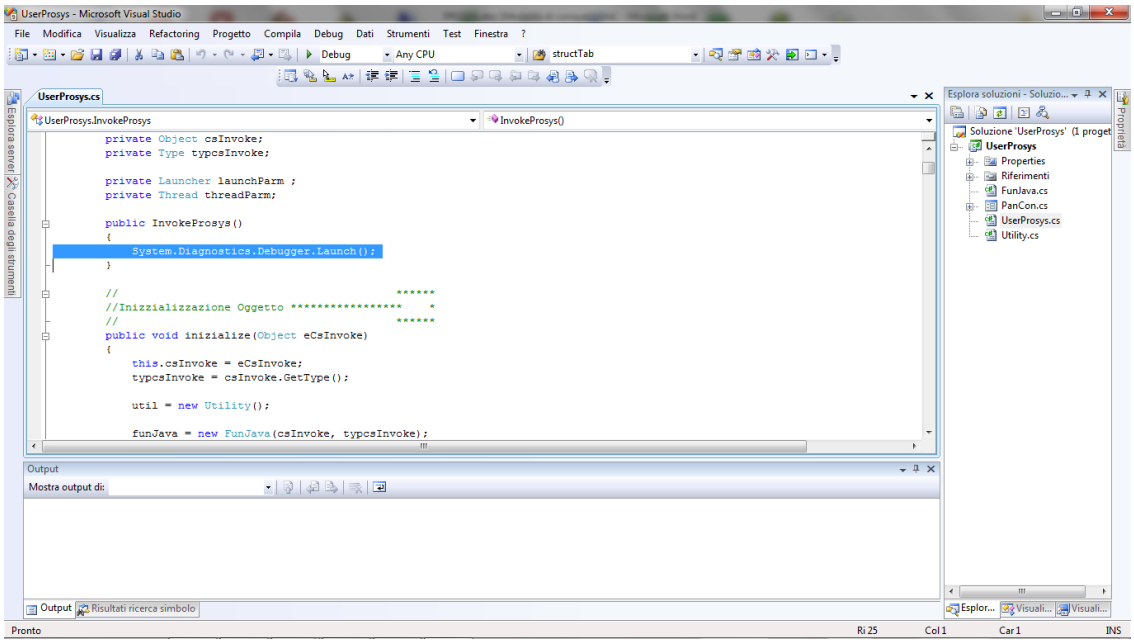


- avvio del progetto;

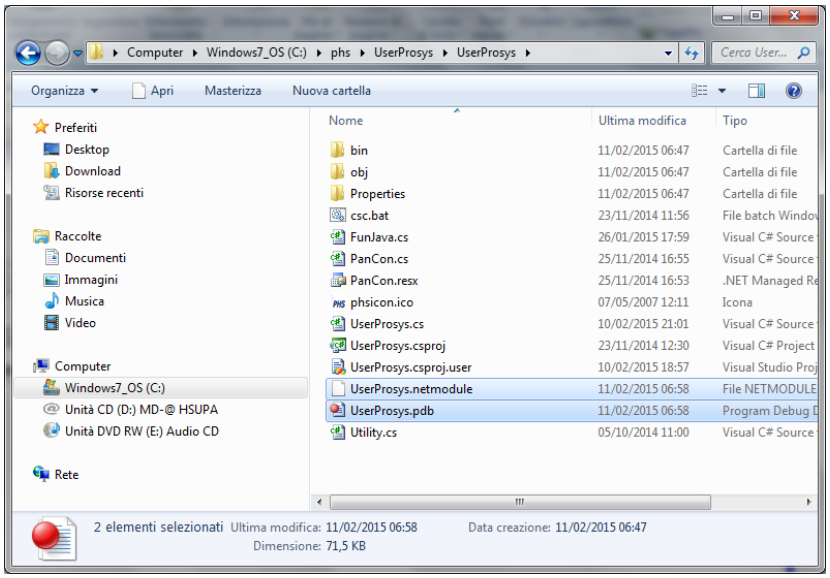


⇐ INDICE

- **decommento** dell'istruzione che permette il lancio del debug;



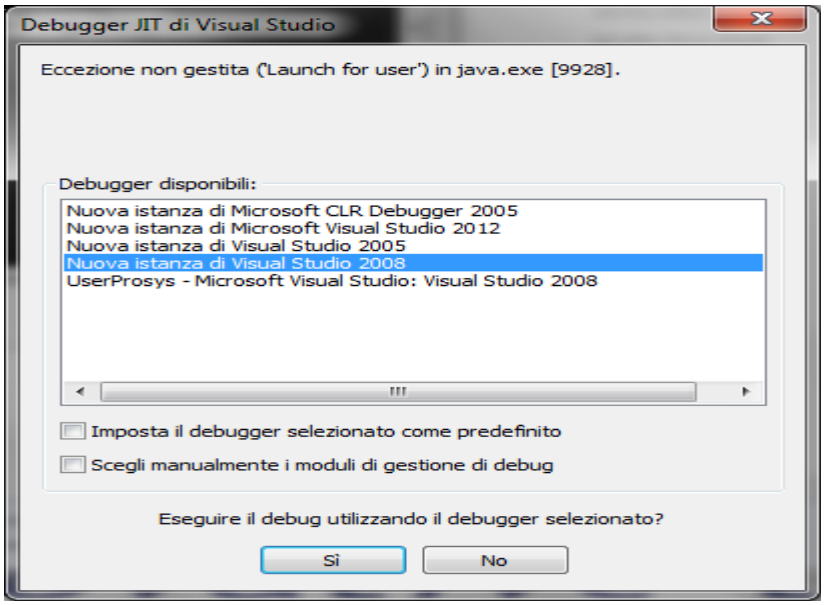
- compilazione, con il file **csc.bat**, dei file **.cs** ottenendo: **UserProsis.netmodule** ed **UserProsis.pdb**;



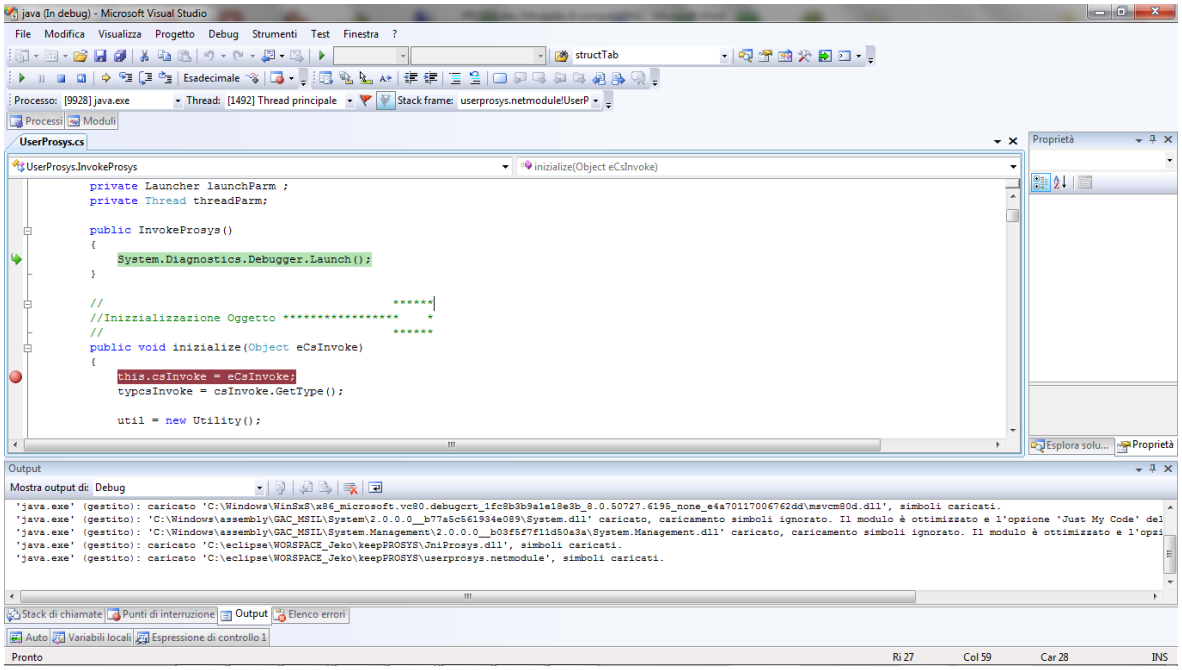
⇐ INDICE

- copia dei file **UserProsis.netmodule** ed **UserProsis.pdb** nella cartella contenente l'installazione del Prosys ed avvio dell'applicazione**keepProsys** (per comodità conviene **usare keepProsys_consolle.exe**); si otterrà:

-

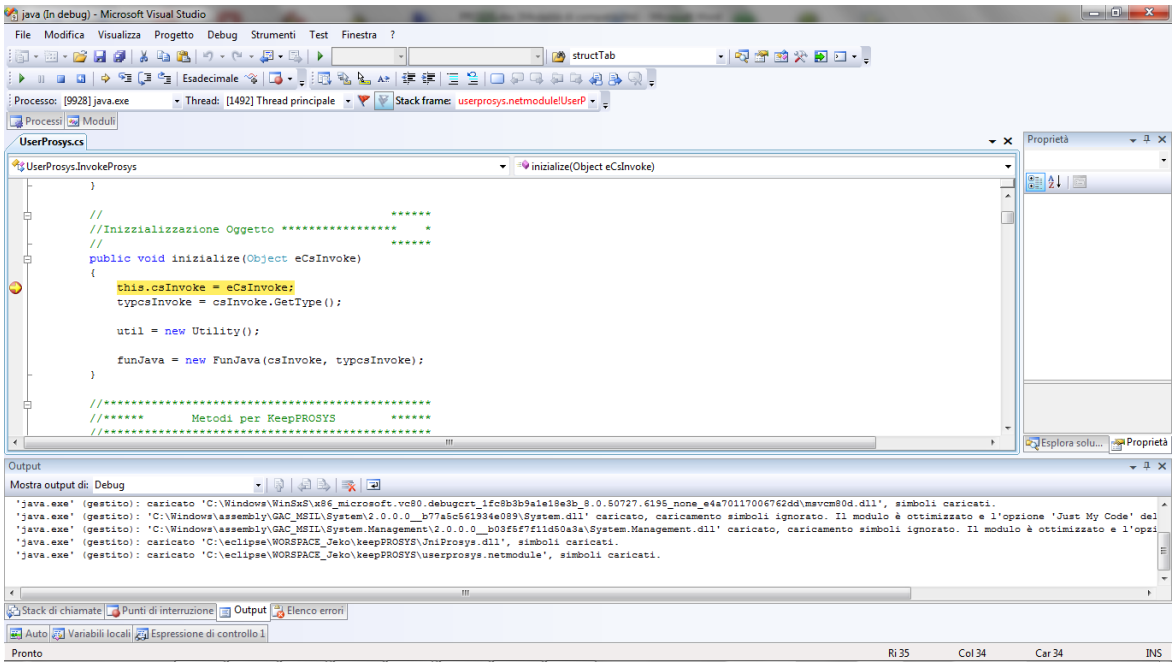


- selezione del debugger ed avvio;



⇐ INDICE

- inseririmento del breakpoint e continuazione;



I metodi della classe **InvokeProsys** vanno sviluppati secondo le proprie esigenze; terminato lo sviluppo,bisogna **commentare** l'istruzione che permette il lancio del debug, compilare con il file **csc.bat** e spostare il file **UserProsis.netmodule** nella cartella contenente l'installazione del Prosys.

⇐ INDICE

TTS_Documentazione

I manuali del sistema TTS sono:

- **TTSSDK**: descrive l'ambiente di sviluppo per il sistema (framework e simulatori);
- **JTTSMAN**: descrive le classi del framework;
- **STREAM**: descrive le stringhe generate e ricevute dall'hardware del TTS.