C/S per l'esecuzione di script JS in Java.

Introduzione:

Il progetto consiste di due software, un Client e un Server scritti in Java in grado di comunicare tra loro nella rete, nello specifico, il Server si occupa di eseguire gli script JavaScript mandati dal Client e rinviarli al medesimo, o in caso di errore di sintassi nello script, inviare quest'ultimo come esito dell'esecuzione.

Download:

Gli eseguibili .Jar, il codice sorgente commentato e la licenza del software, sono scaricabili da questo link.

Per il codice sorgente di base usato per lo sviluppo del progetto, consultare quest'altro link.

Features:

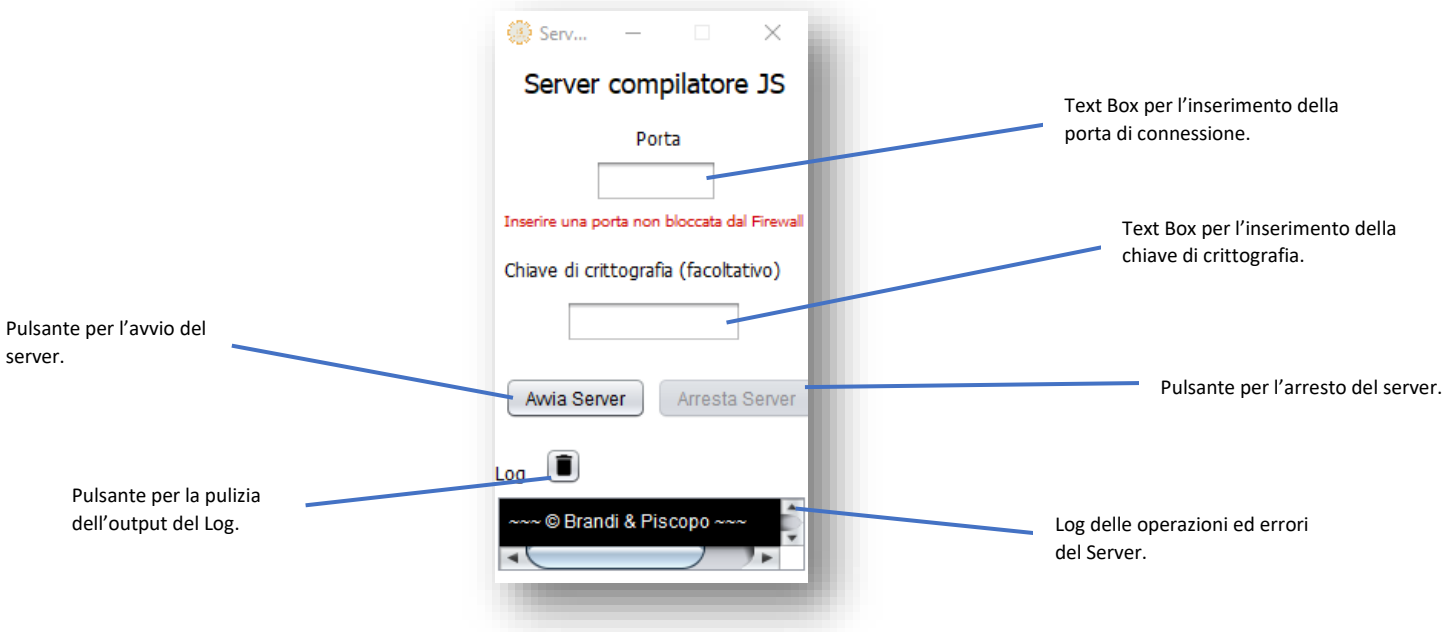
Caratteristiche particolari del programma, paragonato alla sua versione di base sono:

- La presenza di una GUI (Graphical User Interface) gestita su un thread a parte.
- La possibilità di inserire dinamicamente l'indirizzo e/o la porta di comunicazione grazie ai TextBox della GUI.
- La presenza di un controllo sulla validità dell'indirizzo e della porta inseriti.
- La possibilità di criptare la comunicazione.
- La possibilità di arrestare arbitrariamente l'esecuzione del server.
- La chiusura dei ServerSocket, ogniqualvolta ci sia un errore o si arresti volontariamente il server, che permette di liberare la porta, e quindi, di riutilizzarla per una nuova connessione.

GUI:

Il Client e il Server hanno due interfacce grafiche distinte:

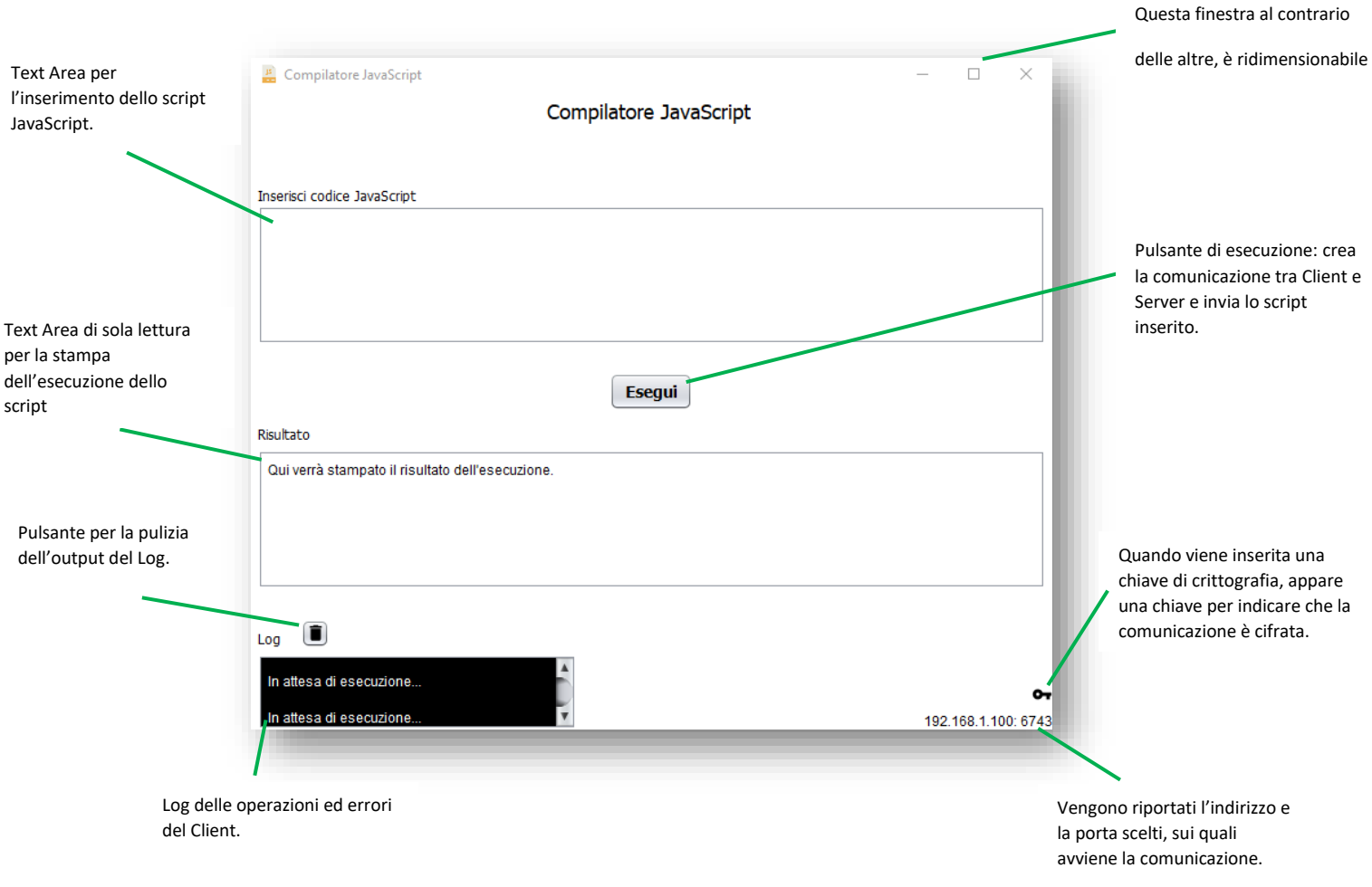
Questa è l'interfaccia del Server:



L’interfaccia del Client consiste, invece, di due schermate. La prima per l’inserimento dei parametri di connessione e della chiave di crittografia, la seconda per l’inserimento e ricezione dello script JS.

Questa è la prima interfaccia del Client:

Questa è la seconda interfaccia del Client:

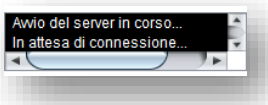


Icona del Client:

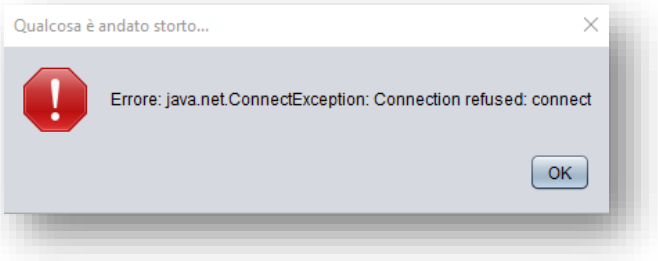
Icona del Server:

Interazione con l'utente:

Durante l'utilizzo del software l'utente è costantemente aggiornato delle operazioni eseguite, poiché per ogni operazione (lettura effettuata/in attesa/connesso/disconnesso) viene effettuata una stampa sul Log.

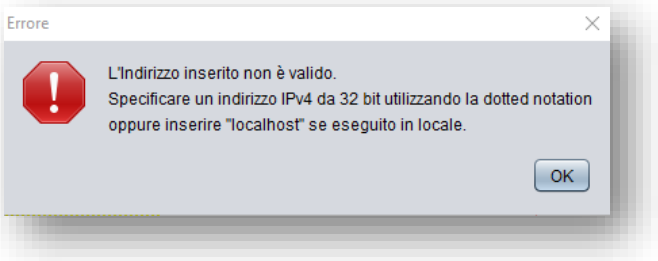


L'utente viene anche avvertito in caso di errori, sia attraverso il Log, sia (nel caso del Client) attraverso messaggi di dialogo.



Inoltre, vengono stampati messaggi di dialogo anche per mancate immissioni o immissioni non valide. Per esempio viene segnalato se la porta ha un valore superiore a 65535 o inferiore a 1024 (0-1023, ovvero le porte riservate, dette "WellKnown"). Viene inoltre verificata la corretta immissione dell'indirizzo, che può essere "localhost", se eseguito in locale, oppure un indirizzo IPv4 da 32 bit (con opportuna dotted notation).

Indirizzo inserito: 192168.1.10

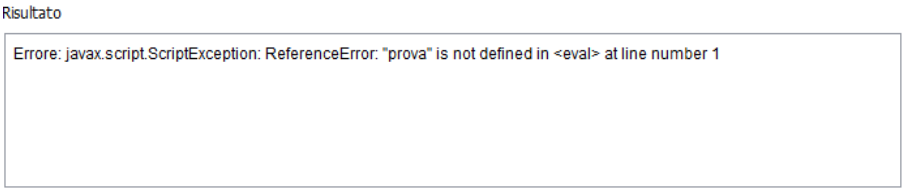


Esecuzione degli script JS:

Come già menzionato, gli script vengono scritti lato client ed eseguiti lato server. Per informazioni circa l'implementazione leggere il paragrafo "pacchetti e classi utilizzati" oppure consultare il codice sorgente interamente commentato. L'esito viene stampato in una Text Area di sola lettura sull'interfaccia del client.

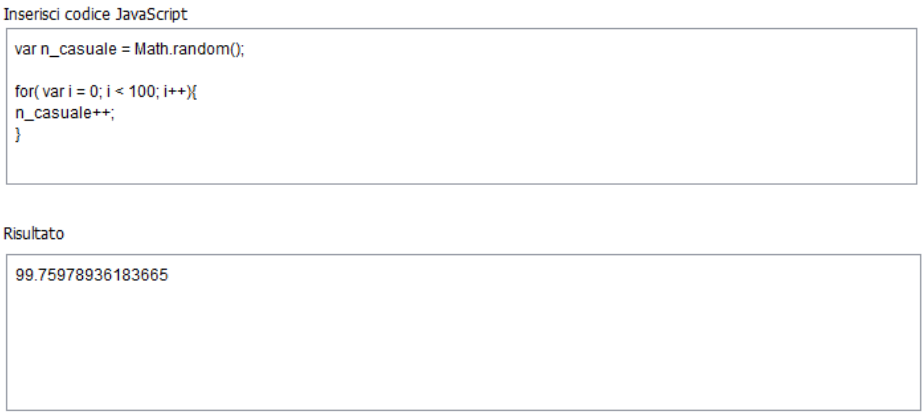
Nel caso in cui venisse inserito uno script non eseguibile (Es: errata sintassi), nella Text Area verrà stampato l'errore.

In questo esempio, è stato scritto "prova" come input: non è un comando JS.



In quest'altro esempio, lo script è stato scritto correttamente e viene stampato il suo esito. È importante notare come il risultato riportato corrisponda all'ultima operazione/assegnazione di variabile, poiché in JavaScript non esiste un metodo di stampa default vero e proprio.

(Infatti non è possibile usare le funzioni di stampa su HTML come document.write(), né altre funzioni come window.alert() o console.log().)



Tipo di applicazione, IDE, e JDK utilizzato:

Per la creazione del software è stato usato Apache NetBeans IDE 11.2, l'applicazione è di tipo JavaFX, e il JDK utilizzato è l'1.8 scaricabile su questo link.

Pacchetti e classi utilizzati:

Per la realizzazione del software del Client e del Server sono stati utilizzati i seguenti pacchetti:

- java.net: per permettere la creazione e la gestione dei socket di connessione.
- java.io: per permettere la creazione e la gestione dei flussi di I/O.
- java.awt e javax.swing per la realizzazione e gestione della GUI.

Inoltre, il Server per poter eseguire gli script JS inviati dal Client, sfrutta le classi ScriptEngine e ScriptEngineManager contenute nel pacchetto javax.script.

Entrambi i software, infine, utilizzano classi presenti nei pacchetti org.json.simple e org.json.simple.parser, contenuti nel json-simple.jar importato nella libreria, per poter comunicare mediante stringhe JSON.

Threading:

Essendo le istruzioni di connessione e di lettura bloccanti per il sistema ed essendo gli oggetti Swing non **thread-safe**, l'esecuzione del codice e la gestione della GUI avvengono su thread diversi. In questo modo è possibile interagire con l'interfaccia grafica anche quando il codice è in esecuzione e/o quando si sta

eseguendo un metodo bloccante. In questo modo, è possibile non solo arrestare il server durante la sua esecuzione, ma anche chiuderne il socket e liberare così, la porta occupata. Per maggiori informazioni, consultare il codice sorgente commentato.

Crittografia:

È possibile, a discrezione dell'utente, cifrare la comunicazione mediante una crittografia simmetrica utilizzante l'operatore logico XOR. Questo particolare tipo di crittografia permette di riutilizzare lo stesso algoritmo (in questo caso richiamato attraverso una funzione), sia per la crittografia che per la decrittografia. Per maggiori informazioni circa l'implementazione di tale algoritmo, consultare il codice sorgente.

Licenza e distribuzione:

L'unico canale di distribuzione del software è GitHub. Il software utilizza la licenza Apache License 2.0 consultabile su questo link.