Il materiale che troverai in questo documento è completamente gratuito. Per quanto riguarda il contenuto, dato che non è stato rivisto da "esperti", mi preme precisare che NON garantisco in alcun modo la correttezza totale.

Lo scopo di tutto ciò non è tanto arricchire il mio curriculum personale, bensì condividere le mie conoscenze con il maggiore numero di persone possibile. Considera questo documento anche come un invito a pubblicare e condividere i tuoi saperi, perché il fondamento di una vera formazione è il libero accesso al sapere in tutte le sue forme.

"Information is power. But like all power, there are those who want to keep it for themselves.

...

Those with access to these resources - students, librarians, scientists - you have been given a privilege.

...

Meanwhile, those who have been locked out are not standing idly by. You have been sneaking through holes and climbing over fences, liberating the information locked up by the publishers and sharing them with your friends. But all of this action goes on in the dark, hidden underground. ***It's called stealing or piracy, as if sharing a wealth of knowledge were the moral equivalent of plundering a ship and murdering its crew.*** *But sharing isn't immoral - it's a moral imperative. Only those blinded by greed would refuse to let a friend make a copy.*

...

There is no justice in following unjust laws. It's time to come into the light and, in the grand tradition of civil disobedience, declare our opposition to this private theft of public culture.

...

With enough of us, around the world, we'll not just send a strong message opposing the privatization of knowledge - we'll make it a thing of the past. Will you join us?"

Guerilla Open Access Manifesto
Aaron Swartz

Progetto Programmazione Java
Servizio di Messaggistica Istantanea

Rambod Rahmani
rambodrahmani@autistici.org
Relentlessly pursue knowledge and selflessly share it with others.

June 8, 2016

Abstract

Il presente documento contiene la documentazione prodotta durante la realizzazione del Progetto "Servizio di Messaggistica Istantanea" per l'esame di "Programmazione Java" - Prof. Mario G.C.A. Cimino - presso l'università degli studi di Pisa - Ingegneria Informatica.
Il progetto Java è disponibile open source su GitHub: https://github.com/rambodrahmani/esame-programmazione-java

Keywords: programmazione , java , esame , unipi , rambod , rahmani

Dedicato a mia madre Sara, mio padre Magid e mio fratello gemello Ramtin per esserci stati quando ho avuto bisogno e per avermi reso la persona che sono oggi.

Indice

1 Introduzione

Il progetto che ho sviluppato per l'esame di Programmazione Java è un software di messaggistica istantanea che permetto di scambio tra due Client connessi contemporaneamente. Il progetto comprende anche un Server di Log che riceve messaggi di Log, relativi all'utilizzo della GUI, da parte dei vari Client connessi al servizio di messaggistica.

Il documento si sviluppa secondo le varie fasi di sviluppo del Progetto.
Le fasi di lavoro sono state preimpostate dalle consegne d'esame pubblicate dal professore.

Il codice sorgente del prodotto e gli altri File prodotti durante lo sviluppo sono disponibili al seguente indirizzo: GitHub - https://github.com/rambodrahmani/esame-programmazione-java

2 Analisi

Il progetto consiste nella realizzazione di un applicativo che permetta lo scambio di messaggi di testo.

Il progetto è composto da:

- **Server di Log**: che si occupa di ricevere i logs, relativi alla navigazione nell'interfaccia utente, dai Clients. Il Server di Log valida il singolo log XML ricevuto dal Client e lo salva in un file di testo (file di log aperto);
- **Client**: Il client è la parte del progetto che permette lo scambio di messaggi di testo. Dotato di interfaccia grafica, permette di visualizzare i Contatti connessi, aprire nuove finestre di conversazione e accedere al diagramma a torta UML.

2.1 Server di Log

Il Server di Log non ha interfaccia grafica. Si occupa di ricevere e conservare, su file di testo locale, i vari logs XML dai Clients connessi.

Vista Dinamica

1. All'avvio, il Server di Log carica da file di configurazione locale i parametri presenti.
2. Una volta avviato il Server, l'applicativo rimane in attesa di connessioni da parte dei vari Clients;
3. Valida i singoli logs XML quando ricevuti dai Clients;
4. Appende su file di testo locale (file di log aperto) i singoli XML validati;
5. Alla chiusura, disconnette i vari Clients connessi e termina.

File di Configurazione Locale in XML
All'avvio del sistema, il Server carica dal file di configurazione locale:

1. La porta da utilizzare per avviare il Server;
2. La locazione del file di Log locale;
3. La locazione del file XSD da utilizzare per validare i singoli log.

Messaggio di Log
Ogni messaggio di Log contiene:

1. L'indirizzo email del Contatto che lo ha inviato;

2. Il tipo di evento;
3. Data e ora dell'invio da parte del Contatto.

File di Log di Testo
Il Sistema riceve e registra un log per i seguenti eventi:

1. Eventi relativi all'utilizzo dell'interfaccia grafica, inviati da parte dei Client.

2.2 Client

Il Client ha interfaccia grafica. Permette allâĂŹutente di visualizzare i Contatti connessi, aprire nuove conversazioni con questi, inviare e ricevere messaggi ed accedere al diagramma a Torta nella finestra Statistiche.

Vista Dinamica

1. All'avvio dell'applicativo, il Client legge dal file di configurazione locale i parametri presenti.
2. Quando l'utente preme il pulsante âĂIJConnettiâĂİ, il Client si connette al Server di Log e si registra nella Base di Dati con l'email inserita;
3. Il Client legge da Base di Dati i Contatti connessi e li carica nella Tabella âĂIJContattiâĂİ;
4. Il Client attende le connessioni da parte dei vari Contatti;
5. A intervalli di tempo determinati, il Client aggiorna la lista Contatti connessi;
6. Lo stato di connessione viene aggiornato impostando il testo della Stringa âĂIJStatoâĂİ che si trova sotto la lista dei Contatti;
7. L'utente puÃš aprire una nuova finestra di conversazione con un Contatto facendo doppio Click sul nome del contatto nella lista Contatti;
8. La finestra di conversazione aperta permette lo scambio di messaggi con il Contatto selezionato;
9. L'utente puÃš utilizzare il menu âĂIJStrumentiâĂİ per accedere alla finestra âĂIJStatisticheâĂİ;
10. La finestra statistiche contiene un diagramma a torta che rappresenta le ore di attivitÃă dei 5 Contatti piÃź connessi.
11. Alla chiusura del Client, il Contatto viene rimosso dalla Base di Dati, si disconnette dal Server di Log e termina.

File di Configurazione Locale in XML
All'avvio del sistema, il Client carica dal file di configurazione locale:

1. Indirizzo e porta del Server di Log a cui connettersi;
2. La porta su cui avviare il Client;
3. L'indirizzo, la porta, il nome utente e la password da utilizzare per accedere al Database.
4. Il numero di ultimi messaggi di ciascuna conversazioni da salvare in Cache.

Cache Locale
Alla chiusura di una finestra di conversazione con un Contatto, il Sistema salva su file binario gli ultimi N messaggi scambiati. All'avvio della successiva conversazione con il Contatto, carica dal suddetto file gli ultimi N messaggi.

Base di Dati
Il Sistema archivia le seguenti informazioni su base di dati:

1. Connessione e disconnessione dei vari Client (email, indirizzo IP, data e ora);
2. I messaggi inviati dai vari Client (mittente, destinatario, messggio, data e ora).

Struttura Tabelle Base di Dati

Contatti

```
contatti (email, indirizzo_ip, data)
email: VARCHAR(45), PRIMARY KEY,
indirizzo_ip: VARCHAR(45),
data: TIMESTAMP
```

La tabella Contatti contiene gli indirizzi Email dei vari Clients connessi e l'indirizzo IP a questi associato.
Ogni contatto è caratterizzato da un indirizzo Email, da un indirizzo IP e dall'ora in cui si è connesso.
Non è possibile che due o piè contatti con lo stesso indirizzo Email possano connettersi.

Messaggi

```
messaggi (id, mittente, destinatario, testo, data)
id: INT(45), PRIMARY KEY,
mittente: VARCHAR(45),
destinatario: VARCHAR(45),
testo: MEDIUMTEXT,
data: TIMESTAMP
```

La tabella Messaggi contiene i messaggi scambiati tra i vari Contatti.
Ogni messaggio è composto da un id numerico (ho preferito utilizzare una colonna chiave dedicata perchè altrimenti si dovrebbe coinvolgere tutte le colonne della tabella per ottenere una chiave univoca), l'indirizzo email del mittente, l'indirizzo email del destinatario, il testo del messaggio e la data e l'ora in cui è stato spedito.

Storico

```
storico (email, durata)
email: VARCHAR(45), PRIMARY KEY,
durata: DOUBLE
```

La tabella Storico contiene, per ciascun Client, l'email e il numero di ore spese connesso. Viene utilizzata dai vari Clients per generare il diagramma a Torta.
Ogni volta che un Client si disconnette, viene utilizzata la colonna data della tabella contatti per calcolare il numero di ore che il Client è rimasto connesso.

File di Log remoto
Il Sistema invia un log per i seguenti eventi:

1. Quando l'utente preme il pulsante "Connetti" per avviare il Client;
2. Quando l'utente preme il pulsante "Disconnetti" per fermare il Client;
3. Quando l'utente accede al menu "Strumenti";
4. Quando l'utente accede al sotto-menu "Statistiche" (Strumenti âĘŠ Statistiche) aprendo il frame Statistiche;
5. Quando l'utente apre una nuova finestra di conversazione con doppio click sulla tabella "Contatti" del Client;
6. Quando un Client invia un messaggio premendo sul pulsante "Invia" del Frame conversazione.

Segue l'interfaccia grafica realizzata per il Client:

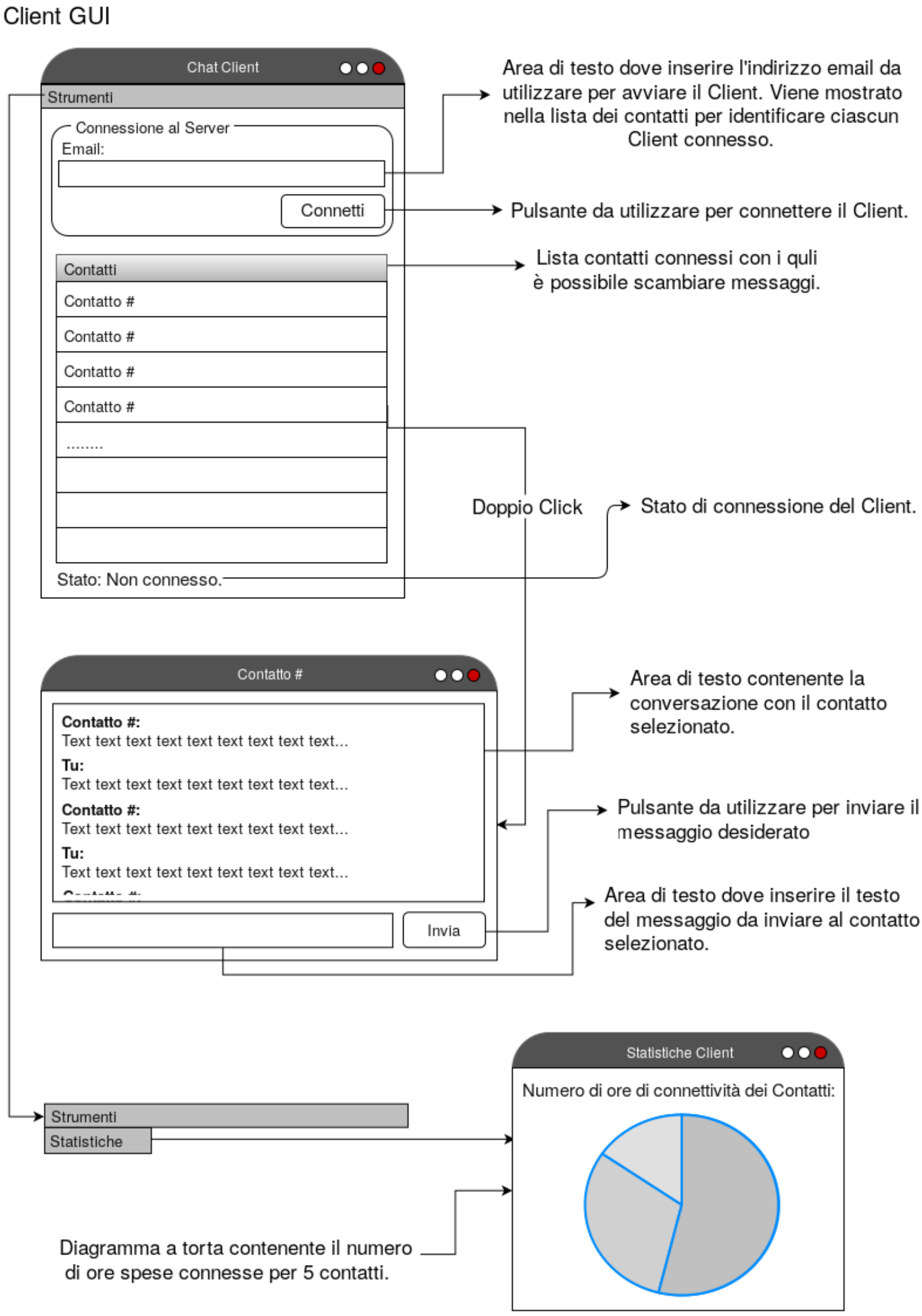

Figura 1: Interfaccia Grafica Client

3 Progetto

3.1 Server di Log

Le classi del Server di Log, come mostrate graficamente nel diagramma UML, sono le seguenti:

Classe Server
Classe contenente la funzione main() del progetto.
Gestisce l'Input/Output da terminale per la gestione (avvio, arresto, stampa logs) del Server di Log.

Classe GestoreParametriConfigurazioneXML
Legge i parametri di configurazione dal file di configurazione .xml locale.
Valida i parametri tramite XML Schema con file .xsd locale.

Classe ParametriConfigurazione
Contiene i parametri di configurazione letti dal file di configurazione .xml locale dalla classe GestoreParametriConfigurazioneXML.
Serializzata/Deserializzata tramite XMLStream.

Classe GestoreLogsXML
Gestisce l'accesso sincronizzato di lettura e scrittura sul file di log .txt locale.
Valida i log ricevuti tramite XML Schema con file .xsd locale.

Classe ThreadServerSocket
Gestisce il ServerSocket in attesa delle connessioni da parte dei Clients.
Estende Thread.
Inviato alla classe SocketClient.

Classe SocketClient
Gestisce il Socket di comunicazione tra Server di Log e ciascun client Connesso.
Estende Thread.

Diagramma UML

3.2 Client

Le classi del Client, come mostrate graficamente nel diagramma UML, sono le seguenti:

Classe Client
Classe contenente la funzione main() del progetto.
Istanzia una variabile di tipo FrameContatti contenente la GUI principale del progetto.

Classe Contatto
Contiene le informazioni di ciascun contatto connesso.
Utilizzata dalla classe GestoreBaseDiDati per ottenere la lista dei contatti connessi.

Classe FrameContatti
Contiene il JFrame con la GUI principale (area connessione, lista contatti connessi, menu strumenti).
Istanzia una variabile di tipo ServerSocketConversazioni che avvia un ServerSocket che rimane in ascolto delle connessione da parte degli altri Contatti.

Classe FrameConversazione
Contiene il JFrame con la GUI della finestra di conversazione.
Istanzia una variabile di tipo SocketConversazione che si connette al ServerSocketConversazioni del contatto con cui si desidera conversare.

Classe FrameStatistiche
Contiene il JFrame con la GUI della finestra delle statistiche.

Classe GestoreBaseDiDati
Esegue tutte le query sulla Base di Dati.

Classe GestoreCacheConversazioni
Si occupa di salvare e prelevare da file binario locale gli ultimi messaggi scambiati con ciascun Contatto.

Classe GestoreParametriConfigurazioneXML
Legge i parametri di configurazione dal file di configurazione .xml locale.
Valida i parametri tramite XML Schema con file .xsd locale.

Classe ParametriConfigurazione
Contiene i parametri di configurazione letti dal file di configurazione .xml locale dalla classe GestoreParametriConfigurazioneXML.
Serializzata/Deserializzata tramite XMLStream.

Classe MessaggioDiLog
Implementa Serializable.
Contiene i campi di ciascun messaggio di log inviato da Client a Server.

Classe ModelloTabellaContatti
Estende AbstractTableModel.
Gestisce i dati della tabella contenente la lista dei Contatti connessi.

Classe ServerSocketConversazioni
Gestisce il ServerSocket in attesa delle connessioni da parte degli altri Contatti.
Estende Thread.
Inviato alla classe SocketConversazione,

Classe SocketConversazione
Gestisce il Socket di comunicazione tra due Clients connessi che scambiano messaggi.
Estende Thread.

Classe SocketServerDiLog
Gestisce il Socket di comunicazione con il Server di Log.
Estende Thread.

Classe Storico
Contiene le informazioni riguardanti il numero di ore di connessione per ciascun Contatto.
Utilizzata dalla classe GestoreBaseDiDati per ottenere la lista delle ore spese connesse per ciascun contatti.

Diagramma UML

3.3 Classi Comuni

Client e Server di Log hanno in comune il package messaggio, che contiene le classi Messaggio e TipoMessaggio, e che verrà sviluppato sotto forma di una Java Class Library, in modo da ottenere un .jar da includere nei due progetti che ne necessitano.
Essendo messaggio l'oggetto serializzato che viene scritto e letto da ObjectOutputStream e ObjectInputStream, conviene avere un unico file .jar da includere nel progetto Server e Client in modo da non incorrere in errori di incongruenza.

Classe Messaggio
Implementa Serializable.
Contiene i campi di ciascun messaggio scambiato tra Server e Client o tra due Clients.

Classe TipoMessaggio
Implementa Serializable.
Enumerazione contenente i tipi di messaggio che possono essere creati.

Diagramma UML

4 Manuale d'uso

Il presente manuale d'uso spiega come utilizzare il Server di Log e il Client del progetto.

Il Server di Log non ha interfaccia grafica e può essere gestito utilizzando la linea di comando, mentre il Client ha un'interfaccia grafica composta da differenti JFrame.

4.1 Server di Log

This statement requires citation [**?**]; this one does too [**?**]. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean dictum lacus sem, ut varius ante dignissim ac. Sed a mi quis lectus feugiat aliquam. Nunc sed vulputate velit. Sed commodo metus vel felis semper, quis rutrum odio vulputate. Donec a elit porttitor, facilisis nisl sit amet, dignissim arcu. Vivamus accumsan pellentesque nulla at euismod. Duis porta rutrum sem, eu facilisis mi varius sed. Suspendisse potenti. Mauris rhoncus neque nisi, ut laoreet augue pretium luctus. Vestibulum sit amet luctus sem, luctus ultrices leo. Aenean vitae sem leo.

4.2 Client

This statement requires citation [**?**]; this one does too [**?**]. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean dictum lacus sem, ut varius ante dignissim ac. Sed a mi quis lectus feugiat aliquam. Nunc sed vulputate velit. Sed commodo metus vel felis semper, quis rutrum odio vulputate. Donec a elit porttitor, facilisis nisl sit amet, dignissim arcu. Vivamus accumsan pellentesque nulla at euismod. Duis porta rutrum sem, eu facilisis mi varius sed. Suspendisse potenti. Mauris rhoncus neque nisi, ut laoreet augue pretium luctus. Vestibulum sit amet luctus sem, luctus ultrices leo. Aenean vitae sem leo.

Bibliografia

1. D. Holmes K. Arnold, J. Gosling. *The Java Programming Language*. Addison-Wesley Professional, 4th edition, 2005.
2. A. Vecchio G. Frosini. *Programmare in Java Vol. I, II, III*. Edizioni ETS, 2005.
3. Oracle Corporation. The Java SE 8 Documentation. `https://docs.oracle.com/javase/8/`, 2014. Accessed: 2016-05-31.
4. Oracle Corporation. The Java FX 2 Documentation. `https://docs.oracle.com/javafx/2/`, 2014. Accessed: 2016-05-31.
5. Oracle Corporation. The Java FX 2 API. `https://docs.oracle.com/javase/8/docs/api/`, 2014. Accessed: 2016-05-31.
6. Oracle Corporation. The Java FX 2 API. `https://docs.oracle.com/javafx/2/api/`, 2014. Accessed: 2016-05-31.