Progetto per il Laboratorio di Sistemi Operativi
Anno Accademico 2019/2020

Stefano Pio Zingaro

May 13, 2020

In questo spazio sono conservati contenuti utili per la documentazione del progetto del corso di Sistemi Operativi per la Laurea Triennale in Informatica per il management dell'Università di Bologna, anno accademico 2019-2020.

I contenuti di questo spazio che riguardano il progetto, sono: 1. Alcune informazioni logistiche utili 2. Una descrizione delle componenti del sistema da sviluppare 3. Le informazioni sulle modalità di consegna dei materiali

Getting Started

In generale, non è necessario usare il contenuto ma solo leggerne le informazioni. In ogni caso è possibile scaricare oppure clonare il codice con il comando `git`:

```
git clone https://github.com/szingaro/lab-so-infoman
```

Build with

- OpenJDK Runtime Environment (build 13+33)
- Jolie 1.9.0 (C) 2006-2020 the Jolie team

Licenza

Logistica

In questa sezione sono riportate le istruzioni sulla formazione dei gruppi. Viene inoltre proposto un calendario per la consegna dell'implementazione e del report. Tali informazioni sono soggette a cambiamenti ed a revisioni, ogni modifica viene comunicata attraverso la bacheca di annunci ufficiale del corso su iol.unibo.it ed il forum ufficiale del corso.

In ogni momento e compatibilmente con la disponibilità dei docenti, è possibile prenotare un ricevimento - in modalità virtuale sulla piattaforma Teams - via messaggio di posta elettronica, specificandone la motivazione, all'indirizzo email del tutor.

Formazione dei gruppi

I gruppi sono costituiti da un minimo di quattro (4) ad un massimo di cinque persone (5), coloro che intendono partecipare all'esame comunicano entro e non oltre il **31 Maggio 2020** (pena esclusione dall'esame) la composizione del gruppo di lavoro, **via posta elettronica**, al tutor. Il messaggio deve essere inviato dalla mail istituzionale del referente del gruppo, ha come oggetto **GRUPPO LSO** e contiene:

1. Il nome del gruppo;
2. Una riga per ogni componente: cognome, nome e matricola;
3. Un indirizzo di posta elettronica di riferimento a cui mandare le notifiche

N.B. è incarico del referente trasmettere le eventuali informazioni agli altri membri.

Email di esempio:

Campo	Testo
Oggetto:	GRUPPO LSO
Nome Gruppo:	NOME MOLTO BELLO
Componenti:	Annio Ennio, 123456
	Sinalefe Pina, 234567
	Pannocchia Anna, 345678
	...
Mail Referente:	anna.pannocchia@studio.unibo.it

Chi non riuscisse a trovare un gruppo invia allo stesso indirizzo di posta elettronica un messaggio con oggetto **CERCO GRUPPO LSO**, specificando:

- Cognome, Nome, Matricola, Email;
- Eventuali preferenze legate a tempi di lavoro (si cercherà di costituire gruppi di persone con tempi di lavoro compatibili, nel limite delle possibilità).

Email di esempio:

Campo	Testo
Oggetto:	CERCO GRUPPO LSO
Componenti:	Annio Ennio, 123456

Campo	Testo
Preferenze	A causa di lavoro full-time (in allegato autocertificazione) sono disponibile solo di sera e nei weekend.

Le persone senza un gruppo vengono assegnate il prima possibile senza possibilità di ulteriori modifiche.

Calendario per la consegna

Le date a disposizione per la consegna dell'implementazione e del report, sono:

- le 23.59.59 di **Lunedí 27 Luglio 2020**;
- le 23.59.59 di **Lunedí 28 Settembre 2020**.

La data presa in considerazione per la consegna della parte di implementazione sarà quella di creazione del **Tag** su GitLab le istruzioni sulla consegna si trovano nelle specifiche per la consegna.

Calendario per l'esame orale

Solo in seguito alle consegne, vengono fissate data ed orario della discussione (notificate alla mail del referente), **entro una settimana dalla data di consegna**.

La discussione dell'implementazione e la demo di funzionamento vengono effettuate in un incontro *virtuale* con tutti i componenti. Al termine della discussione, ad ogni singolo componente viene assegnato un voto in base all'effettivo contributo dimostrato nel lavoro. Altre informazioni nella specifiche per la consegna.

Il progetto

L'architettura decentralizzata peer-to-peer

Il sistema di messaggistica che vogliamo realizzare comprende un'architettura di rete paritaria e un sistema software basato su servizi. Come far coesistere questi due paradigmi?

I sistemi *peer-to-peer* (*P2P*) prevedono la comunicazione tra nodi in una rete paritaria - dove tutte le entità sono pari tra loro. Un nodo in una rete P2P viene detto *peer* e si occupa contemporaneamente di offrire e richiedere servizi agli altri *peer* della rete.

Un sistema software è implementato come un'architettura di servizi se ogni sua componente è un servizio. I servizi richiedono operazioni esposte da altri servizi

attraverso delle opportune interfacce che identificano alcune informazioni importanti per la riuscita della comunicazione, quali ad esempio il nome dell'operazione, il tipo della richiesta e quello della risposta (in caso ce ne fosse una).

Per permettere ai *peer* di scambiarsi messaggi abbiamo bisogno che ognuno di essi esponga una serie di operazioni per:

- registrarsi alla rete presso un altro *peer* ed ottenere così un identificativo — uno *pseudonimo*.
- scrivere un messaggio in un file locale
- pubblicare un messaggio per un altro/altri *peer*

Pubblicazione e scrittura dei messaggi

Implementare un sistema di messaggistica istantanea tra nodi di una rete può essere affrontato in molti modi diversi. Uno di questi modi - uno dei più interessanti nella nostra prospettiva "concorrente" - è quello di vedere il messaggio come una richiesta di scrittura su una risorsa condivisa da molti processi (ricorda niente?).

Il nostro caso particolare si può ricondurre ad un problema di gestione del sistema di **pubblicazione** e **scrittura** su file di un messaggio. L'argomento è trattato in questo *blog post*, con particolare attenzione ai problemi che si possono incontrare nell'implementazione Jolie.

Comunicazione tra *peer* nella rete

I *peer* comunicano in due modalità: si scambiano messaggi privati e pubblicano messaggi su canali - creati dagli stessi *peer* - dove tutti possono scrivere.

Nella prima delle due modalità - quella che prevede uno scambio di messaggi tra *mittente* e *destinatario* - il sistema assicura che il messaggio non possa essere letto da terzi, criptando la comunicazione. Nella seconda modalità - la scrittura su chat pubblica - il sistema garantisce sia l'identità del mittente che l'integrità del contenuto dei messaggi al momento della consegna.

Messaggi privati cifrati

Alice e Bob vogliono comunicare - in privato - scambiandosi messaggi l'uno con l'altra, per farlo in sicurezza hanno deciso di usare un protocollo di crittografia asimmetrica:

1. Bob cripta il messaggio con la chiave pubblica di Alice;
2. Alice decripta il messaggio di Bob con la sua chiave Privata.

Ovviamente, nel caso in cui sia Alice a voler inviare un messaggio a Bob, ella eseguirà gli stessi passaggi di Bob, il quale invece farà come aveva fatto Alice.

Figure 1: crittografia asimmetrica

Canali pubblici con messaggi firmati

Alice ha bisogno di comunicare ad un gruppo di persone un qualche messaggio e decide di farlo apponendo una firma "digitale" al suo messaggio cosicchè tutto i destinatari possano controllare che nessun altro ha cambiato il contenuto di quello che Alice voleva comunicare e che è stata proprio lei, e non un altro, a scrivere il messaggio.

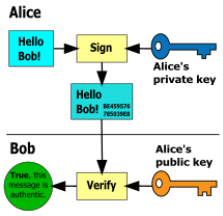

Figure 2: firma digitale asimmetrica

1. Alice crea una cosidetta impronta del suo messaggio, applicando una funzione di *hash* ad esso e producendo una stringa
2. Alice cifra il risultato dell'hash con la sua chiave privata, il risultato dell codifica è la firma digitale di Alice per quel messaggio
3. Alice allega la firma digitale al messaggio in chiaro e invia tutto ai suoi destinatari
4. Bob e chiunque altro riceva il messaggio, controlla che l'hash (lo stesso algoritmo di Alice) del messaggio in chiaro corrisponda con il risultato della decifratura della firma; quest'ultima fatta con la chiave pubblica di Alice.

Monitoraggio della rete

L'ultimo componente - ma non per questo meno importante - del nostro sistema, è il *monitor* di rete. Esso si occupa di trascrivere (indifferentemente in un file oppure su console) il log di sistema. Ogni operazione invocata nella rete deve, subito prima o subito dopo, inviare un messaggio di log al *monitor*.

La sua interfaccia dovrebbe apparire simile alla seguente:

```
interface IMonitor {
    RequestResponse:
        log( string )( void )
}
```

Dove, per semplicità, il tipo della richiesta dell'operazioe `log` è una semplice stringa. L'uso di una operazione di tipo `RequestResponse`, che implementa una comunicazione bloccante vuole forzare il "chiamante" di questa operazione ad attendere finchè il *monitor* non ha ultimato la scrittura sulla coda dei *logs*. L'esempio è a titolo puramente esemplificativo e non costituisce alcun obbligo nella scelta della soluzione pià appropriata per questo componente.

Specifiche per la consegna

Globalmente, vengono consegnati due prodotti: la documentazione e l'implementazione (il codice sorgente) del progetto. La valutazione finale avviene mediante una esame orale, nella quale vengono discusse le strategie con la quale i prodotti consegnati sono stati generati. Le domande saranno rivolte a tutti i partecipanti al gruppo, i quali potranno scegliere l'esposizione organizzata dei contenuti.

- non si accettano richieste di eccezioni sui progetti con motivazioni legate a esigenze di laurearsi
- chi copia o fa copiare invalida il progetto - il codice viene controllato con un programma per il rilevamento di plagio - si rimanda alla pagina del tutor per maggiori informazioni sul codice d'onore del laboratorio.

La documentazione

È possibile scrivere la documentazione nel formato preferito, l'importante è che il PDF generato rispetti la struttura del modello (più avanti). La documentazione ha lunghezza di **almeno** quattro o cinque pagine (quindi da 8 a 10 facciate), è scritto con font di grandezza *12pt* e viene consegnato in formato PDF. Il limite è un *lower bound*; non esiste un *upper bound* per la lunghezza del report che può quindi essere di un numero arbitrario di pagine.

Di seguito viene riportato un esempio di documentazione con le principali caratteristiche da inserire.

L'intestazione della Documentazione

- Laboratorio Sistemi Operativi A.A. 2019-2020
- Nome del Gruppo
- Indirizzo mail del referente: nome.cognome@studio.unibo.it
- Componenti:
  - Cognome, Nome, Matricola
  - ...

Il corpo della Documentazione

1. Descrizione generale del progetto
    1. Componenti del progetto
    2. Funzionalità implementate
    3. Contenuto della documentazione
2. Istruzioni per la demo - le istruzioni per eseguire una demo.
3. Discussione sulle strategie di implementazione:
    1. Struttura del progetto - come è stato diviso il progetto tra i componenti del grupppo. Quali sono i problemi principali riscontrati nell'implementazione? Quali sono le alternative considerate? Perché sono state adottate alcune soluzinoi piuttosto di altre?
    2. Sezione di descrizione della - ad esempio, abbiamo implementato l'operazione `foo` perché abbiamo bisogno di ... Il codice di esempio (riporto il codice) mostra il problema appena descritto ... risolto con ...

In ogni caso motivare le proprie scelte giustificando le proprie azioni.

Griglia di Valutazione per la documentazione

La valutazione della documentazione verte sull'analisi dello scritto e sulla sua capacità di esprimere con chiarezza i concetti descritti, soprattutto **grazie all'uso di esempi**.

In particolare la griglia di valutazione usata è la seguente:

Criterio	Descrizione
Qualità dell'informazione	Riconoscimento dei problemi (di concorrenza) e loro descrizione
Uso degli esempi	Presenza di almeno un esempio in tutte le scelte implementative
Analisi delle scelte implementative	Descrizione della propria scelta implementativa e presenza di proposte di alternative valide

L'implementazione del progetto

Il progetto viene sviluppato utilizzando il linguaggio Jolie. Non ci sono requisiti riguardo ai protocolli *protocolli* e i *media* utilizzati per realizzare la comunicazione tra i componenti del sistema.

La gestione del progetto avviene col supporto del sistema `git`, a questa pagina è possibile trovare una lista di comandi utili per imparare ad usare questo strumento di scrittura collaborativa.

Il codice del progetto è contenuto in uno spazio in cloud del servizio online

GitLab e gestito seguendo la procedura descritta di seguito.

N.B. Le istruzioni che seguono devono essere completate entro e non oltre la scadenza di presentazione dei gruppi.

1. Ogni membro del gruppo crea un account su GitLab o accede con le proprie credenziali
2. Il **referente del gruppo** crea un nuovo progetto cliccando sul bottone + in alto a destra nella schermata principale di GitLab, inserendo `LabSO_NomeGruppo` come nome del progetto e cliccando su **New Project**
   - una volta che il progetto è stato creato, il referente aggiunge ogni membro del gruppo con `role permission` > `Developer` andando su `Settings` > `Members` nel menù a sinistra, cercandoli in base allo username con il quale questi si sono registrati su GitLab;
   - il referente aggiunge l'utente `stefanopiozingaro` con `role permission` > `Reporter`.

La consegna dell'implementazione

Al momento della consegna, il repository dovrà contenere i sorgenti del progetto e la relazione, nominata `REPORT_LSO.pdf`.

Per effettuare la consegna:

1. Nella pagina del progetto, cliccare sulla voce del menù `Repository` > `Tags` > `New Tag`
2. Digitare come `Tag Name` il nome **Consegna**
3. Cliccare su `Create Tag` per eseguire la creazione del `Tag` di consegna

Una volta creato il Tag, inviare una email di notifica di consegna con soggetto **CONSEGNA LSO - NOME GRUPPO** a stefanopio.zingaro@unibo.it.

Griglia di Valutazione per l'implementazione

La valutazione dell'implementazione del sistema si basa sull'analisi del codice Jolie, sull'uso dei costrutti del linguaggio per la creazione di soluzioni efficienti, sulla tolleranza ai guasti del sistema implementato e sulla gestione delle eccezioni.

In particolare la griglia di valutazione usata è la seguente:

Criterio	Descrizione
Uso dei costrutti di Jolie	Corretto utilizzo dei costrutti per la gestione della concorrenza, uso corretto di `execution(...)`

Criterio	Descrizione
Distribuzione del carico di lavoro nel gruppo	Omogeneità nella ripartizione dei compiti nel gruppo, ogni membro partecipa egualmente allo sviluppo indicando il singolo contributo, assenza di dissimmetria di informazione.
Grado di partecipazione alla comunità	Presenza di domande e risposte significative sui canali di comunicazione offerti dal corso, richieste di ricevimento e delucidazioni, bug fixing del materiale messo a disposizione del docente e dalla comunità *open source* di Jolie

La dimostrazione (demo)

Insieme alla documentazione ed al codice sorgente, dovrà essere preparato uno script che permette di automatizzare i test. Almeno nella fase iniziale della prova orale, può essere utile preparare del materiale da correlare allo script (ad esempio *screenshot*), che permetteranno di velocizzare le operazioni di controllo del codice. Tale suite di test *può* essere intergrata nel codice sorgente (è tuttavia opzionale).