UNIVESITÁ DEGLI STUDI DI MILANO
LAUREA TRIENNALE IN COMUNICAZIONE DIGITALE
PROGETTO LABORATORIO DI RETI DI CALCOLATORI

News sharing

Matteo Zignani

21 dicembre 2013

1 PRESENTAZIONE DEL PROBLEMA

Un elemento che caratterizza le odierne online social network è sicuramente la possibilità di condividere vari tipi di contenuti con i propri amici: dai video fino alla notizie che reputiamo interessanti. Alcuni di questi servizi ci permettono di specificare con quali amici condividere i contenuti evitando di intasare le bacheche dei rimanenti amici con messaggi e video scarsamente utili e interessanti. Lo scopo del progetto di questo appello è la creazione di un servizio social minimale che permette di inoltrare in modo selettivo le notizie. Un utente registrato a questo servizio ha la possibilità di richiedere le notizie specificando a quale categoria (selezionabile tra un insieme predefinito di valori) è interessato. Il servizio invia all'utente un insieme di notizie e l'utente ha la possibilità di inoltrare quest'ultime agli amici. La notizia viene sempre inoltrata all'amico/i selezionato/i ma viene considerata come pubblicabile sul profilo dell'utente solo se la categoria della notizia appartiene alle categorie ammissibili definite per l'utente destinatario. Ogni utente, infatti, seleziona, al momento della creazione del profilo, un insieme di categorie di interesse (selezionabili tra lo stesso insieme predefinito di categorie valido per le notizie). Una notizia pubblicabile può essere postata sulla bacheca di Facebook accompagnata da un emoticon di commento che indica se la notizia è positiva, negativa oppure neutra a seconda del sentimento che essa suscita nel lettore.
Il servizio che lo studente deve implementare si basa su un'architettura client/server. Il server deve gestire la rete sociale tra gli utenti iscritti e deve reperire le notizie da una fonte specifica che verrà illustrata nelle prossime sezioni. Il server, inoltre, mette a disposizione delle risorse richiedibili utilizzando il protocollo HTTP che permettono la gestione, la diffusione e la pubblicazione delle notizie, secondo i vincoli specificati in seguito. Per agevolare la pub-

blicazione dei post su Facebook è stato creato un utente fittizio di nome Giovanni Algoritmo sul cui profilo è possibili postare le notizie e i commenti.
Lo studente deve sviluppare solamente la parte server dell'architettura descritta. Per quanto riguarda il client si possono utilizzare vari programmi per la fase di test: telnet (Linux e Mac), putty () o un qualsiasi browser facendo richieste a localhost:<portaServerInAscolto>. Il server DEVE essere MULTITHREAD.

2 API

Nelle seguenti sottosezioni vengono descritti le componenti del server e le risorse che esso mette a disposizione.

2.1 Formato della richiesta

Il server riconosce come valide le richieste che seguono il protocollo HTTP. In particolare il server riconosce solo richieste con metodo di accesso di tipo GET. Per questo motivo lo studente NON deve implementare un server HTTP completo ma un suo piccolo sottoinsieme. Un esempio di possibile richiesta valida può essere:

```
GET <URL Risorsa con eventuali parametri> HTTP/1.0
user-agent: Mozilla 5.0, Chrome, Safari
host: localhost:6000
content-type: text/html
<rigaVuota>
```

Il server deve verificare la validità della riga di intestazione e ignorare tutti i campi settati nella definizione dell'header. Una volta rilevata la stringa vuota di terminazione della richiesta, il server deve eseguire l'operazione richiesta dall'URL e rispondere secondo il formato HTTP. Si noti che l'URL della riga di intestazione della richiesta deve seguire la sintassi specifica di un URL (vedi slide corso). Se la richiesta è andata a buon fine, il formato della risposta è:

```
HTTP/1.0 200 OK

<messaggio>
```

dove < $messaggio$ > è un messaggio testuale che dipende dalla risorsa invocata. Nel caso di un qualsiasi errore il formato della risposta sarà:

```
HTTP/1.0 400 ERRORE

<messaggio>
```

Anche in questo caso < $messaggio$ > contiene un messaggio di errore dipendente dalla risorsa invocata.

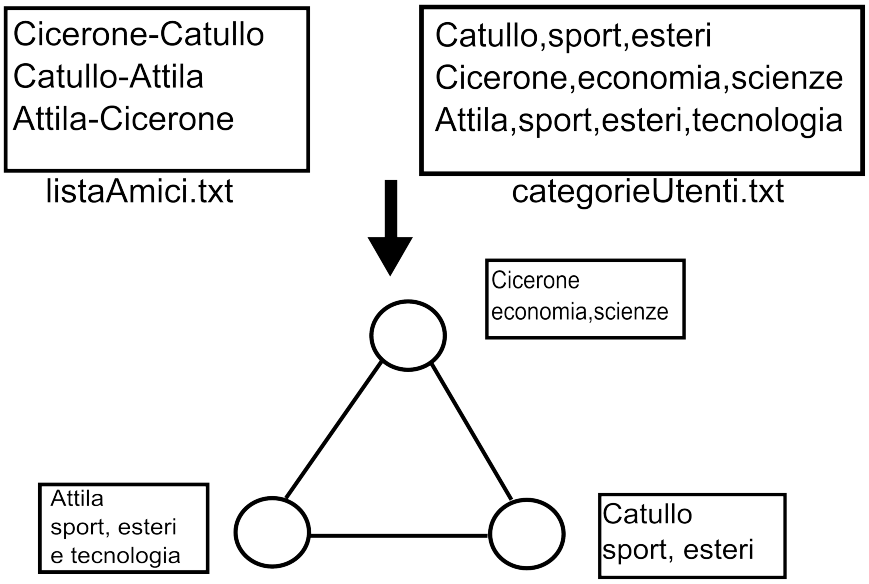

Figura 2.1: Esempio costruzione della rete sociale.

2.2 CREAZIONE DELLA SOCIAL NETWORK

Per quanto riguarda la rete sociale che impone i vincoli sulla diffusione delle notizie, essa viene caricata al momento della creazione del servizio. Gli utenti e i relativi nodi non possono essere modificati, mentre è possibile creare degli archi durante l'esecuzione del servizio. Di conseguenza, non si possono aggiungere o eliminare utenti ma è possibile creare nuove relazioni di amicizia tra gli utenti presenti, oltre a quelle presenti al momento dell'inizializzazione del server. Il caricamento avviene per mezzo di un file chiamato *listaAmici.txt* che contiene una lista degli archi non orientati presenti nella rete sociale. Ogni riga del file rappresenta un arco e segue il formato:

```
<nomeUtente>-<nomeUtente>
```

dove < $nomeUtente$ > rappresenta un segnaposto (< e > non vanno inseriti). Lo studente deve prevedere una classe per il caricamento della social network da un file. Il file *listaAmici.txt* si trova sul sito del corso di Reti come materiale di supporto al progetto.
Un secondo file (*categorieUtenti.txt*) contiene, per ogni utente, le categorie a cui l'utente è interessato. Ogni riga del file di testo descrive un utente e segue il formato:

```
<nomeUtente>,<categoria1>,<categoria2>,<categoriaN>
```

Al momento del caricamento del grafo, lo studente deve prevedere anche l'assegnamento delle categorie agli utenti. Nella figura 2.1 viene presentato un piccolo di grafo costruito partendo dai due file.

2.3 AGGIUNGERE UNA RELAZIONE DI AMICIZIA

Un utente può aggiungere un amico alla sua cerchia utilizzando la risorsa $aggiungiAmico$. La risorsa *aggiungiAmico* accetta come unici parametri validi $nomeUtente$ e $nomeAmico$, il primo indica l'utente che vuole creare l'amicizia, mentre il secondo indica il nome (stringa) dell'utente verso cui creare una relazione di amicizia. L'arco che viene creato è non orientato e non necessita che l'utente destinatario della richiesta accetti la creazione. Un possibile esempio di richiesta valida è il seguente:

```
GET /aggiungiAmico?nomeAmico=Cicerone&nomeUtente=Seneca /HTTP/1.0
user-agent: Mozilla 5.0, Chrome, Safari
host: localhost:6000
content-type: text/html
<rigaVuota>
```

Una possibile risposta positiva è data da:

```
HTTP/1.0 200 OK

Amicizia con Cicerone creata con successo
```

Possibili errori da gestire sono l'assenza del nodo di destinazione/sorgente della richiesta o la richiesta di creazione di una relazione che già esiste.

2.4 LISTA DEGLI AMICI

Un utente può visualizzare i propri amici utilizzando la risorsa $vediAmici$. La risorsa necessita dell'argomento $nomeUtente$: l'utente che richiede la lista degli amici. Nel corpo della risposta, $vediAmici$ restituisce un array JSON contenente una serie di oggetti JSON che rappresentano gli amici. Ogni oggetto JSON è costituito da due campi: $idUtente$ contenente il nome utente dell'amico e $categorie$ contenente un array JSON di stringhe, ognuna indicante una categoria di interesse dell'amico. Un possibile esempio di richiesta valida è la seguente:

```
GET /vediAmici?nomeUtente=Seneca /HTTP/1.0
user-agent: Mozilla 5.0, Chrome, Safari
host: localhost:6000
content-type: text/html
<rigaVuota>
```

Una possibile risposta positiva è data da:

```
HTTP/1.0 200 OK

[
{"idUtente": "Cicerone", "categorie": ["economia" , "scienze"]},
{"idUtente": "Attila", "categorie": ["sport" , "esteri" , "tecnologia"]}},
```

```
{"idUtente": "Batiato", "categorie": ["cronaca" , "esteri"]}},
{"idUtente": "Enea", "categorie": ["politica" , "cronaca" , "esteri"]}}
]
```

Possibili errori sono costituiti dall'errata sintassi della risorsa.

2.5 RICHIESTA DELLE NOTIZIE

Un utente può ricevere le notizie attraverso la risorsa *prendiNotizie* . Questa risorsa necessita dell'argomento *categorie*, in cui si deve specificare una lista di categorie separate dalla virgola e dell'argomento *nomeUtente*, l'utente che richiede le notizie. La risorsa *prendiNotizie* restituisce un array JSON di oggetti JSON, in cui ogni oggetto rappresenta una notizia. Ogni notizia è caratterizzata da tre campi: *idNotizia* (un intero che identifica univocamente la notizia), *titolo* (titolo della notizia) e *categoria* (categoria della notizia). Le categoria delle notizie restituite deve appartenere alla lista delle categorie specificata nella richiesta. Il numero di notizie per categoria e il numero totale di notizie restituite è lasciato a discrezione dello studente. Un possibile esempio di richiesta di notizie è il seguente:

```
GET /prendiNotizie?categorie=sport,scienze,cronaca&nomeUtente=Seneca /HTTP/1.0
user-agent: Mozilla 5.0, Chrome, Safari
host: localhost:6000
content-type: text/html
<rigaVuota>
```

Una possibile risposta positiva è data da:

```
HTTP/1.0 200 OK

[
{"idNotizia": 1, "titolo": "Uno, due, Tevez", "categoria": "sport"},
{"idNotizia": 2, "titolo": "Scoperta una villa romana a Roma", "categoria": "scienze"},
{"idNotizia": 5, "titolo": "Attila era vegetariano", "categoria": "scienze"},
{"idNotizia": 100, "titolo": "Weekend tra i maiali", "categoria": "cronaca"}
]
```

Possibili errori possono nascere da elementi nella lista delle categorie che non appartengono all'insieme delle categorie valida. **L'insieme delle categorie valide è costituito dalle stringhe** *economia*, *politica*, *esteri*, *tecnologia*, *cronaca*, *sport* **e** *scienze*.

2.5.1 RICHIESTA AL REPOSITORY DELLE NOTIZIE

Il server che gestisce il servizio di news sharing deve reperire le notizie dal web server in ascolto sulla porta 80 all'indirizzo 159.149.133.70. La risorsa che restituisce le notizie è chiamata *getNotizie* e non riceve alcun argomento. La risorsa risponde con un array JSON che segue lo stesso formato della risposta restituita dalla risorsa *prendiNotizie*. Se la richiesta è errata viene restituito un array JSON vuoto.

2.6 DIFFUSIONE DELLE NOTIZIE AGLI AMICI

Un utente può suggerire e diffondere una notizia ad un determinato amico per mezzo della risorsa *diffondiNotizia*. La risorsa prevede tre parametri: *nomeUtente*, *nomeAmico* e *idNotizia*. Il secondo parametro rappresenta il nome dell'amico a cui *nomeUtente* vuole inviare la notizia indicata dal terzo parametro. Si noti che la risorsa diffonde la notizia all'amico SOLO se la categoria della notizia appartiene alle categorie di interesse dell'amico. Un possibile esempio di richiesta è la seguente:

```
GET /diffondiNotizia?nomeUtente=Senece&nomeAmico=Catullo&idNotizia=1 /HTTP/1.0
user-agent: Mozilla 5.0, Chrome, Safari
host: localhost:6000
content-type: text/html
<rigaVuota>
```

Una possibile risposta positiva è data da:

```
HTTP/1.0 200 OK

Notizia 1 inoltrata con successo a Catullo
```

Possibili errori possono essere dovuti alla mancata presenza del nome utente, alla mancata relazione di amicizia con l'utente specificato, alla mancanza di una notizia con l'id specificato oppure alla categoria non presente nella lista delle preferenze dell'amico.

2.7 RICHIESTE DELLE NOTIZIE INVIATE DAGLI AMICI

Un utente può richiedere l'elenco delle notizie che gli sono state inviate usando la risorsa *richiediNotizie*. La risorsa richiede il solo parametro *nomeUtente* e restituisce come risposta un array JSON nel formato specificato per la risorsa *prendiNotizie*. Un possibile esempio di richiesta è la seguente:

```
GET /richiediNotizie /HTTP/1.0
user-agent: Mozilla 5.0, Chrome, Safari
host: localhost:6000
content-type: text/html
<rigaVuota>
```

2.8 PUBBLICAZIONE DELLE NOTIZIE DIFFUSE

L'utente può pubblicare una notizia sul profilo di un utente definito dall'id **100007294060146** utilizzando la risorsa *pubblicaNotizia*. Questa risorsa richiede tre parametri: *nomeUtente*, *idNotizia* e *reazione*. *idNotizia* indica quale notizia *nomeUtente* vuole pubblicare sul profilo, mentre *reazione* indica il sentimento che suscita la notizia. L'ultimo campo può assumere tre valori: *positivo*, *negativo* e *neutro*. La risorsa pubblica un post sulla bacheca dell'utente 100007294060146 il cui messaggio equivale al titolo della notizia con id *idNotizia* e un commento al post pubblicato il cui messaggio è un emoticon. Se *reazione* è

uguale a positivo viene pubblicata la stringa :-), se negativo la stringa :-(, se neutro la string :-|. **Un utente può pubblicare solo notizie che gli sono state diffuse dai suoi amici.** Per esempio il risultato sul profilo dell'utente alla richiesta:

```
GET /pubblicaNotizia?idNotizia=1&reazione=positivo /HTTP/1.0
user-agent: Mozilla 5.0, Chrome, Safari
host: localhost:6000
content-type: text/html
<rigaVuota>
```

è mostrato nella seguente figura:

Il server inoltre risponde al client inviando una risposta HTTP il cui corpo contiene un messaggio di avvenuta pubblicazione, o meno, del post e del commento. L'access token da utilizzare per la pubblicazione è il seguente:
CAADaUwpF1ZAEBAF82vZAVpGRmX6eydVUUAp0RGwndROZCmIZBvZC1sPFtviwnwL0d9Fbf3DQfetluTQ3WVZBXqEqxizUZC8BQX6NvpO3uArlGXgzaZBKJVr1GVzelqceKoxOFm3j6yPwU3iRW1n1qi8ZCKQXJTbCmSno3XqYQ3U6NnaEARvBsDYH7KgI84R8BNcsZD

3 Modalità di consegna

Il progetto è individuale e verranno presi provvedimenti nel caso ci sia il sospetto di una eventuale copiatura. In particolare non verranno ammessi alla discussione del progetto e quindi saranno automaticamente esclusi dall'appello quei progetti con almeno un file con indice di similarità superiore al 70% (moss, tool di Stanford). Questo per non incentivare progetti scritti singolarmente ma frutto di brain-storming troppo di gruppo.
Il linguaggio da utilizzare per lo sviluppo è Java. Il codice del progetto e i relativi jar delle librerie utilizzate devono essere inviati al docente in una directory compressa (zip o tar.gz) denominata 'progettoFebbraio2014'. La mail deve avere come oggetto 'ProgettoFebbraio2014' e contenere nel corpo nome, cognome e numero di matricola dello studente. Il materiale deve essere consegnato entro le **23:59:59** del **20 gennaio 2014**. Valgono tassativamente le seguenti regole:

- Il codice che non compila non verrà considerato
- I progetti consegnati dopo la data di consegna non saranno ritenuti validi

Per essere considerato sufficiente e quindi ammissibile per la discussione, il progetto deve implementare correttamente tutte le funzioni precedentemente illustrate. Principali criteri di valutazione del codice:

- Gestione di possibili errori nell'input da parte dell'utente
- Modularità del codice e rispetto del paradigma ad oggetti
- Gestione oculata delle eccezioni
- Uso appropriato dei commenti
- Gestione e risoluzione di eventuali problematiche legate alla concorrenza sia lato client sia lato server

Per chiarimenti, dubbi potete contattare il docente all'indirizzo matteo.zignani@unimi.it, specificando come oggetto obbligatorio 'ProgettoReti'. Eventuali modifiche al testo del progetto saranno comunicate sulla pagina web del corso.