Progetto di laboratorio – Programmazione III - 2018/2019

Si sviluppi un'applicazione java che implementi un servizio di posta elettronica organizzato con un mail server che gestisce le caselle di posta elettronica degli utenti e i mail client necessari per permettere agli utenti di accedere alle proprie caselle di posta. Si assuma di avere 3 utenti di posta elettronica che comunicano tra loro.

- Il mail server gestisce una lista di caselle di posta elettronica e ne mantiene la persistenza utilizzando file (txt o binari, a vostra scelta) per memorizzare i messaggi in modo permanente.
- Il mail server ha un'interfaccia grafica sulla quale viene visualizzato il log delle azioni effettuate dai mail clients e degli eventi che occorrono durante l'interazione tra i client e il server.
    - Per esempio: apertura/chiusura di una connessione tra mail client e server, invio di messaggi da parte di un client, ricezione di messaggi da parte di un client, errori nella consegna di messaggi, eliminazione di messaggi, etc. (tutte le tipologie di azioni permesse dai client
    - *NB: NON fare log di eventi locali al client come per esempio il fatto che ha schiacciato un bottone, aperto una finestra o simili in quanto non sono di pertinenza del server.*
- Una casella di posta elettronica contiene:
    - Nome dell'account di mail associato alla casella postale (es. giorgio@mia.mail.com).
    - Lista eventualmente vuota di messaggi. I messaggi di posta elettronica sono istanze di una classe Email che specifica ID, mittente, destinatario/i, argomento, testo e data di spedizione del messaggio.
- Il mail client, associato a un particolare account di posta elettronica, ha un'interfaccia grafica così caratterizzata:
    - L'interfaccia permette di:
        - creare e inviare un messaggio a uno o più destinatari (destinatari multipli di un solo messaggio di posta elettronica)
        - leggere i messaggi della casella di posta
        - rispondere a un messaggio ricevuto, in Reply (al mittente del destinatario) e/o in Reply-all (al mittente e a tutti i destinatari del messaggio ricevuto)
        - girare (forward) un messaggio a uno o più account di posta elettronica
        - rimuovere un messaggio dalla casella di posta.
    - L'interfaccia mostra sempre la lista aggiornata dei messaggi in casella e, quando arriva un nuovo messaggio, notifica l'utente attraverso una finestra di dialogo.
    - NB: per semplicità si associno i mail client agli utenti a priori: non si richiede che il mail client offra le funzionalità di registrazione di un account di posta. Inoltre, un mail client è associato a una sola casella di posta elettronica e la sua interfaccia non richiede autenticazione da parte dell'utente.
- *NB: il mail client non deve andare in crash se il mail server viene spento – gestire i problemi di connessione al mail server inviando opportuni messaggi di errore all'utente.*

Requisiti tecnici:

- L'applicazione deve essere sviluppata **in Java** e basata su **architettura MVC**, con Controller + viste e Model, seguendo i principi del pattern **Observer Observable**. Si noti che non deve esserci comunicazione diretta tra viste e model: ogni tipo di comunicazione tra questi due livelli deve essere mediato dal controller o supportata dal pattern Observer Observable.
- L'applicazione deve permettere all'utente di correggere eventuali input errati (per es., in caso di inserimento di indirizzi di posta elettronica non esistenti, il server deve inviare messaggio di errore al client che ha inviato il messaggio).
- I client e il server dell'applicazione devono **parallelizzare le attività** che non necessitano di esecuzione sequenziale e gestire gli eventuali problemi di accesso a risorse in mutua esclusione. NB: i client e il server di mail devono essere applicazioni java distinte; la creazione/gestione dei messaggi deve avvenire in parallelo alla ricezione di altri messaggi.
- L'applicazione deve essere **distribuita** (i mail client e il server devono stare tutti su JVM distinte) attraverso l'uso di Socket Java.

Requisiti dell'interfaccia utente:

- L'interfaccia utente deve essere:
    - Comprensibile (**trasparenza**). In particolare, a fronte di errori, deve segnalare il problema all'utente.
    - Ragionevolmente efficiente per permettere all'utente di eseguire le operazioni con un numero minimo di click e di inserimenti di dati.
    - Deve essere implementata utilizzando **JavaFXML** e, se necessario, **Thread java.** Non è richiesto l'uso di Java Beans, properties e binding di properties.

NOTE:

- Si raccomanda di prestare molta attenzione alla progettazione dell'applicazione per facilitare il parallelismo nell'esecuzione delle istruzioni e la distribuzione su JVM diverse.
- *Si ricorda che:*
    - *Il progetto può essere svolto in gruppo (max 4 persone) o individualmente. Se lo si svolge in gruppo la discussione deve essere fatta dall'intero gruppo in soluzione unica.*
    - *La discussione potrà essere fatta nelle date di appello orale del corso oppure su appuntamento, concordando via email la data con il docente a cui il gruppo è assegnato.*
    - *Si può discutere il progetto prima o dopo aver sostenuto la prova scritta.*
    - *Come da regolamento d'esame il voto finale si ottiene come media del voto dello scritto e di quello di laboratorio (i due voti hanno ugual peso nella media).*
    - *Il voto finale deve essere registrato entro fine settembre 2019, data oltre la quale non è possibile mantenere i voti parziali.*

Leggere il regolamento d'esame sulla pagina web del corso per ulteriori dettagli.