Progetto di laboratorio di Programmazione III - versione completa

Il Progetto di laboratorio consiste, complessivamente di **due applicazioni (progetti)** distinte:

1) Un **mail server** che gestisce le caselle di posta elettronica degli utenti registrati sul server;

2) Un **mail client** che ciascun utente può eseguire per leggere la propria posta elettronica, inviare email ad altri account di posta elettronica (e/o a se stesso), etc.

- **Il mail client e il mail server** sono entrambi implementati come **applicazioni javaFXML basate sul pattern MVC. Ciascuno di essi è implementato in un progetto java distinto e organizzato in package** per modularità.
    - All'interno di queste applicazioni **non deve esserci comunicazione diretta tra viste e model:** ogni comunicazione tra questi due livelli deve essere mediata dal controller o supportata dal pattern Observer Observable.
    - **Non si usino le classi deprecate Observer.java e Observable.java.** Si usino le classi di JavaFX che supportano il pattern Observer Observable.
- **Le applicazioni client e server girano in Java Virtual Machine distinte** e comunicano solo ed esclusivamente attraverso la **trasmissione di dati testuali in socket Java.**
- **Il mail client e il mail server devono parallelizzare le attività che non necessitano di esecuzione sequenziale e gestire gli eventuali problemi di accesso a risorse in mutua esclusione.** Si raccomanda di prestare molta attenzione alla progettazione per facilitare il parallelismo nell'esecuzione delle istruzioni.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

MAIL CLIENT - SPECIFICHE:

- Quando l'utente lancia il mail client, il client chiede di inserire l'indirizzo di posta elettronica e utilizza questo come identificatore dell'utente durante l'esecuzione.
- Il mail client mantiene i seguenti dati **durante la propria esecuzione**:
    - **indirizzo di posta elettronica** dell'utente;
    - **casella postale dell'utente (inbox):** lista dei **messaggi di posta elettronica** ricevuti dall'utente e non cancellati. Non gestite il cestino e neppure la outbox.
- **Il mail client è ignaro di quali siano gli utenti registrati sul server.** Quando l'utente inserisce un indirizzo di posta elettronica, ne verifica la correttezza sintattica (ben formatezza) utilizzando le espressioni regolari (Regex), chiedendo all'utente di reinserirlo se sintatticamente errato.

- Per verificare **se un indirizzo di posta (ben formato) è esistente, il mail client si connette al server.**
- Si assuma che una stessa persona usi sempre lo stesso device per leggere la mail.

GUI (Graphical User Interface):

- **La GUI (view di FXML) gestisce SOLO la INBOX** e permette all'utente di:
  - **inserire il proprio indirizzo di posta elettronica** come unica forma di autenticazione (non è prevista l'iscrizione di un nuovo utente);
  - **visualizzare la lista dei messaggi in entrata** e scorrerla per selezionare il messaggio da visualizzare in dettaglio;
  - **visualizzare i dettagli di uno specifico messaggio** di posta elettronica;
  - **cancellare un messaggio** di posta elettronica dalla inbox;
  - **creare un messaggio di posta elettronica** specificando uno o più destinatari (verificare la correttezza sintattica degli indirizzi di posta elettronica inseriti e permettere all'utente di reinserire i dati se sintatticamente errati); inviare il messaggio al server tramite opportuno bottone;
  - **rispondere a un messaggio di posta elettronica** presente nella inbox in REPLY (solo al mittente) o in REPLY-ALL (a tutti i destinatari del messaggio);
  - **inoltrare un messaggio (forward)** a uno o più destinatari (verificare la correttezza sintattica degli indirizzi di posta elettronica inseriti e permettere di reinserire i dati se sintatticamente errati);
  - **visualizzare lo stato della connessione con il server** (connesso/non connesso).
- **L'interfaccia utente deve essere:**
  - **responsive:** la GUI dovrà mostrare automaticamente la lista dei messaggi aggiornata, senza che l'utente debba compiere azioni specifiche per fare il refresh. Inoltre, all'arrivo di un nuovo messaggio dovrà notificare l'utente.
  - **comprensibile (trasparenza).** Inoltre, a fronte di errori, deve segnalare il problema all'utente.
  - **funzionale (efficacia)** per permettere di eseguire operazioni limitando il numero di click da fare.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

MAIL SERVER:

- **Il mail server gestisce una lista di caselle di posta elettronica** e **ne mantiene la persistenza utilizzando file** (txt o binari, a vostra scelta, non si possono usare database) per memorizzare i messaggi in modo permanente.
- **Ogni 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 server ha un'interfaccia grafica sulla quale viene visualizzato il log 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 a uno o più destinatari, errori nella consegna di messaggi ai destinatari.
  - NB: NON fare log di eventi locali al client come il fatto che ha schiacciato un bottone, aperto una finestra o simili in quanto non sono di pertinenza del server.

- **NB: si assuma che il server abbia un numero fisso di account di posta elettronica, precompilato (per es. 3 account).** Non si richiede che da mail client si possano registrare nuovi account di posta sul server.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

COMUNICAZIONE TRA CLIENT E SERVER:

- **La verifica dell'esistenza degli indirizzi di posta elettronica è responsabilità del server.** In caso l'utente inserisca indirizzi di posta elettronica non esistenti, il server deve inviare messaggio di errore al client. Per esempio, in merito al fallimento di un'autenticazione, oppure in caso l'utente tenti di inviare un messaggio a un account inesistente.
- **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 e fare in modo che il mail client si riconnetta automaticamente al server quando questo è novamente attivo.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

ULTERIORI REQUISITI (MAIL CLIENT E MAIL SERVER)

- Per la dimostrazione si assuma di avere 3 utenti di posta elettronica che comunicano tra loro. **Si progetti però il sistema in modo da renderlo scalabile a molti utenti.**
- **Non gestite socket permanenti per collegare client e server:** fate in modo che, come HTTP, il client chieda di aprire la connessione ogni volta che ha bisogno di fare un'operazione.
- **Non trasferite intere caselle di posta elettronica da client a server, o viceversa,** per questioni di scalabilità del servizio. Quando il client chiede aggiornamenti al server, il server deve solo inviare i messaggi che non sono stati precedentemente distribuiti al client.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

PROMEMORIA PER L'ESAME

- **Il progetto SW può essere svolto in gruppo (max 3 persone) o individualmente.** Se lo si svolge in gruppo la discussione deve essere fatta dall'intero gruppo in soluzione unica. Tenetene conto per formare i gruppi che iniziano lo sviluppo del mail client.
- La discussione potrà essere fatta nelle date di appello orale dell'insegnamento, che saranno distribuite su tutto l'Anno Accademico.
- Si può discutere il progetto SW prima o dopo aver sostenuto la prova orale.
- Come da regolamento d'esame il voto finale si ottiene come media del voto della prova teorica (scritta) e della discussione di laboratorio (i due voti hanno ugual peso nella media).
- Il voto finale deve essere registrato entro fine settembre 2025, data oltre la quale non è possibile mantenere i voti parziali. Leggere il regolamento d'esame sulla pagina web dell'insegnamento per ulteriori dettagli.

Ultime modifiche: lunedì, 11 novembre 2024, 13:53