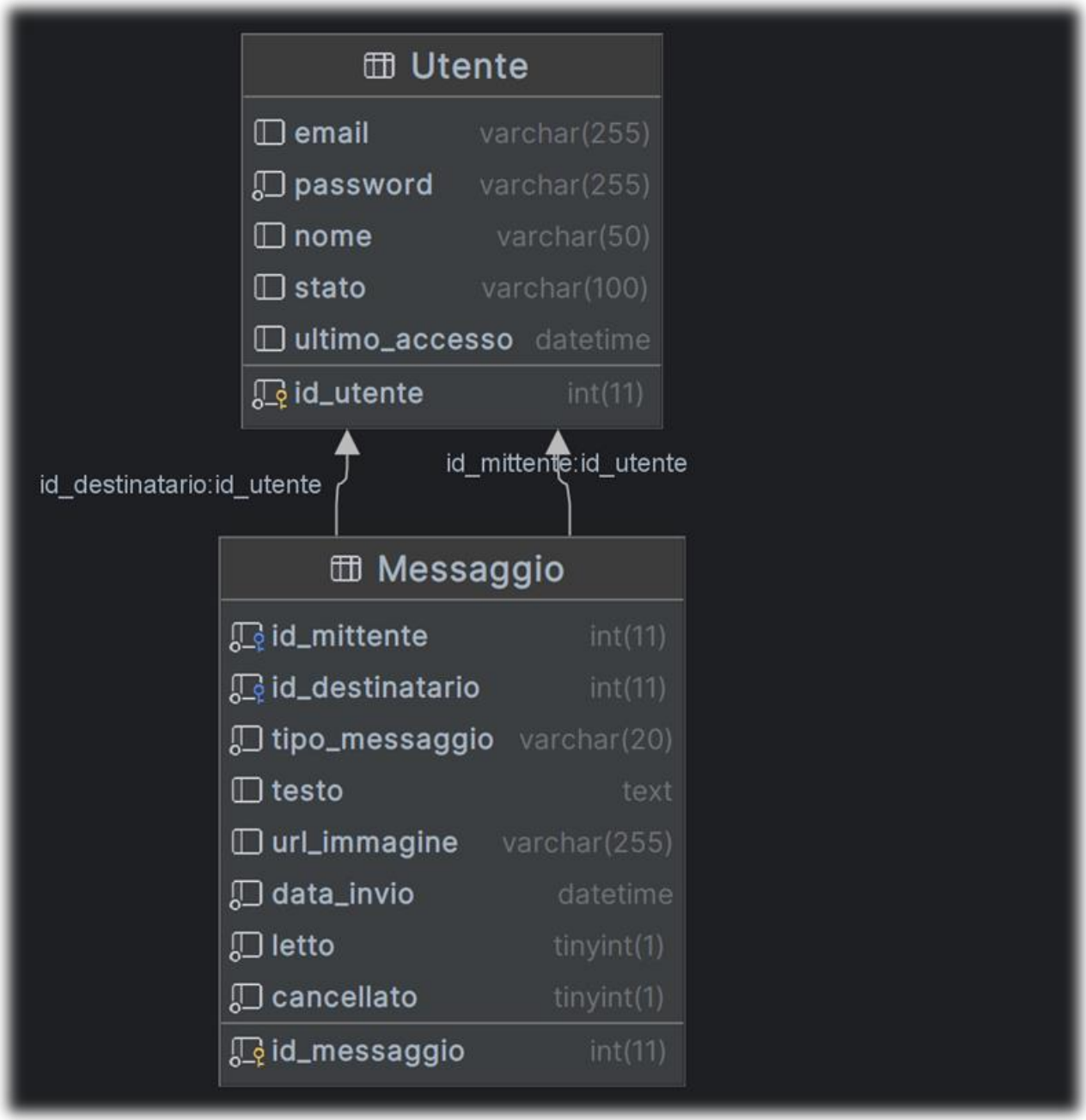
DOCUMENTAZIONE G.C. 5BITI

PARTE 1 – DATABASE

Modello del database (un utente può avere molti messaggi, ogni singolo messaggio ha 2 id_utente, quello del mittente e quello del destinatario).



Di Gabriele Caretti 5BITI – Documentazione per progetto CHAT usando API.

PARTE 2 – JAVABEANS

Sono stati creati i seguenti JavaBeans:

- MessaggioBean
- UtenteBean

MESSAGGIOBEAN

Contiene tutti i metodi per inserire e rimuovere i messaggi dal database.

Ossia:

- getMessaggio(); // Ritorna un messaggio in base all'ID specificato nel bean.
- inserisciMessaggio() // Inserisce un messaggio con i dati del Bean.
- Diversi getter e setter.

UTENTEBEAN

Contiene tutti i metodi per gestire gli utenti (utile per la sessione a seguito dell'autenticazione).

Ossia:

- getUtente(); // Carica l'utente in base all'id impostato.
- getUtentePerEmail(); // Carica l'utente in base all'email impostata.
- getMessaggiInviati(); // Carica i messaggi inviati dell'utente e ritorna un ArrayList di MessaggioBean.
- getMessaggiRicevuti(); // Carica i messaggi ricevuti dall'utente e ritorna un ArrayList di MessaggioBean.
- getMessaggiInviatiERicevuti(); // Carica i messaggi inviati e ricevuti dall'utente e ritorna un ArrayList di MessaggioBean.
- leggiMessaggi(); // Metodo utility per leggere i messaggi usando una struttura standard nel progetto.
- getListaUtentiContattati(); // Ritorna un ArrayList di UtenteBean degli utenti contattati almeno una volta.
- getListaUtentiDaContattare(); // Ritorna un ArrayList di UtenteBean degli utenti mai contattati.
- Diversi getter e setter.

API – REST

Glassfish è particolare e richiede di creare una classe di appoggio che controlli poi una seconda classe con gli URL per la REST.

In questo caso, sono state create le classi:

- Api -> Aggiunge /api all'url prima di ogni chiamata REST ai vari indirizzi disponibili, e ascolta le varie risorse come ChiamateAPI.
- ChiamateAPI è la classe che contiene tutti gli URL per la lettura e inserimento di messaggi e dati tramite REST.

CHIAMATEAPI

Diversi endpoint contattabili, ATTENZIONE: Aggiungere prima /api altrimenti gli indirizzi non verranno trovati!

- / -> Serve per verificare che funzioni la classe, ritorna un hello world.
- /hello/{name} -> Legge {name} con l'input utente e ritorna "hello nome", serve sempre per debug e test.
- /login -> POST -> Richiede 2 parametri, email e password, utilizza DBConnection.autenticazione(email, password) per eseguire l'autenticazione dell'utente, in seguito ritorna true se effettuata con successo, false se non.

- /post/utenti/contatti -> POST -> Tramite id_utente inviato come parametro, legge e ritorna la lista degli utenti contattati almeno una volta. Ritorna un JSON.
- /post/messaggi/ricevuti -> POST -> Tramite id_utente e id_utente_contatto ritorna i messaggi ricevuti dall'id_utente da id_utente_contatto. Ritorna un JSON.
- /post/messaggi/inviati -> POST -> Tramite id_utente e Id_utente_contatto ritorna i messaggi inviati dall'id_utente a id_utente_contatto. Ritorna JSON.
- /post/chat -> POST -> Tramite id_utente e id_utente_contatto ritorna l'intera chat tra i due utenti ordinati dal messaggio più vecchio a quello più recente. Ritorna JSON.
- /post/inviaMessaggio -> POST -> Tramite id_utente, id_utente_contatto e testo, invia un messaggio con mittente id_utente e destinatario id_utente_contatto. Ritorna JSON.
- /post/utentiNonContattati -> POST -> Tramite id_utente, ritorna un JSON contenente tutti gli UtenteBean mai contattati da id_utente. Ritorna JSON-
- /post/nome -> POST -> Tramite id_utente, ritorna l'UtenteBean dell'id_utente richiesto. Ritorna JSON.

Altri endpoints non completati:

- /post/inviaMessaggioImmagine -> POST -> Tramite id_utente, id_utente_contatto, testo e immagine, avrebbe dovuto leggere l'immagine inviata e il testo per poi archiviarli sul server, salvarli nel database e poterci poi riaccedere tramite URL, tuttavia ci sono complicanze con l'invio dei dati dell'immagine.

ALTRE CLASSI:

- DBConnection -> Gestisce la connessione con il database e anche l'autenticazione dell'utente.
- UploadImmagine -> Servlet per il caricamento di una immagine con testo inviata dall'utente, NON FUNZIONANTE.
- ImgUtility, avrebbe dovuto aiutare l'archiviazione dell'immagine, la lettura tramite serializzazione e tutto il necessario, INUTILIZZATA.
- LocalDateTimeAdapter -> Adapter per interpretare in modo corretto i dati dell'ora fino al secondo del database e anche JSON.
- Login -> Servlet di login.
- Logout -> Servlet di logout.

UTILIZZO:

Il progetto deve essere caricato come un progetto maven, successivamente tramite glassfish con le impostazioni di default è possibile avviarlo e usufruire delle API e webservices.

Per testare se i webservices siano funzionanti, collegarsi al seguente indirizzo e verificare che l'hello world venga caricato normalmente:
http://localhost:8080/webservicesglassfish-1.0-SNAPSHOT/api/

LATO UTENTE:

Il sito è accessibile al seguente URL in locale a seguito dell'avvio:
http://localhost:8080/webservicesglassfish-1.0-SNAPSHOT/login.jsp

Sarà poi necessario loggarsi con uno dei due utenti già creati, o entrambi (con una tab in incognito per esempio), successivamente si potrà usare il sito per inviare i messaggi, l'interfaccia è abbastanza intuitiva.

Dopo aver premute da entrambi gli utenti il pulsante "Carica messaggi" si avrà una chat che ogni circa 5 secondi aggiornerà i messaggi inviati. Non è necessario che entrambi gli utenti abbiano "Carica messaggi" attivo per inviare messaggi, questi saranno salvati nel database e visualizzati su richiesta.

Di Gabriele Caretti 5BITI – Documentazione per progetto CHAT usando API.

NOTA: IL CARICAMENTO DI ALLEGATI NON FUNZIONA, QUINDI PER FAVORE EVITARNE L'USO.

Se si crea un nuovo utente e si volesse inviarne un messaggio per la prima volta, bisogna dalla barra in alto premere su "Seleziona un contatto" e selezionarlo, poi premere sul tasto "Scrivi", si aprirà la pagina normale per inviare i messaggi, a seguito dell'invio del primo messaggio, sarà visibile come ogni altro utente nella barra a sinistra.