ODD

Object Design Document

SOMMARIO

1. Introduzione

1.1 Object Design Trade-offs

Trade offs

La seguente sezione descrive eventuali conflitti che sorgono tra due o più design goals. In questo documento si tratteranno soltanto i più significativi.

Comprensibilità vs Costi

Abbiamo sviluppato una documentazione per permettere la comprensione del codice, a persone esterne al progetto. La comprensibilità risulta alquanto elevata, ciò viene facilitato anche grazie all'utilizzo di tecnologie software open source e alle varie spiegazioni.

Prestazioni vs Costi

Per ridurre i costi abbiamo deciso di utilizzare tecnologie open source, le prestazioni non saranno eccessive ma saranno garantite per il carico di lavoro prestabilito così da rendere piacevole l'utilizzo del software.

Costi vs Mantenimento

Abbiamo deciso di voler presentare un software completo, infatti non sono state decise future implementazioni. Sarà quindi garantito un software dalle buone prestazioni.

Interfaccia vs Easy-use

L'interfaccia, è stata implementata in modo da essere facilmente usabile da una vasta gamma di utenti.

Il nostro scopo era creare un'interfaccia con la quale l'utente potesse avere una facile esperienza (Easy-use) con il sistema.

Ogni pulsante chiarisce in maniera accurata il proprio utilizzo, in modo da evitare qualsiasi tipo di ambiguità all'utente.

Interfaccia vs Tempo di efficienza

Abbiamo fatto in modo che il tempo di risposta tra server e interfaccia sia più che sufficiente a soddisfare ogni richiesta dell'utente, senza intaccare in maniera decisiva l'efficienza del sistema.

1.2 Componenti Off-the-Shelf

Nel nostro sistema vengono utilizzati componenti off-the-shelf ovvero componenti software già sviluppati, noti e funzionali per ridurre i tempi e i costi di sviluppo del software. Questi ultimi permettono di ottenere allo stesso tempo effetti grafici potenti, usando il componente Bootstrap, e funzioni avanzate di javascript, usando il componente J-Query.

Bootstrap è un framework per la creazione di siti e applicazioni per il web. Essa contiene modelli di progettazione basati su HTML e CSS, sia per la tipografia che per le varie componenti dell'interfaccia come moduli, pulsanti e navigazione.

J-Query è una libreria JavaScript per applicazioni web. Questa nasce con l'obiettivo di semplificare la selezione, la manipolazione, la gestione degli eventi e l'animazione di elementi DOM in pagine HTML, nonché di implementare funzionalità AJAX.

1.3 Linee Guida per la Documentazione delle Interfacce

Gli sviluppatori, nell'implementare il sistema, devono attenersi alle linee guida di seguito descritte.

1.3.1 CamelCase (notazione a cammello)

È la pratica di scrivere frasi o parole composte, unendo tutte le parole tra loro, in modo tale che ogni parola o abbreviazione nel mezzo della frase inizi con una lettera maiuscola, senza spazi o punteggiatura intermedi. Sono riconosciuti due tipi di annotazione CamelCase: UpperCamelCase, che prevede di scrivere l'iniziale della prima parola composta con lettera maiuscola, e lowerCamelCase, che prevede di scrivere l'iniziale della prima parola composta con lettera minuscola.

1.3.2 Classi, interfacce, pacchetti Java

Per la dicitura CamelCase, far riferimento alla sezione 1.3.1 CamelCase.

- Nomi dei pacchetti: i nomi dei pacchetti sono tutti in minuscolo, con parole consecutive concatenate insieme (senza caratteri di underscores).
  Ad esempio: è corretto questo nome di pacchetto, "com.esempio.platformlearning"; invece sono sbagliati questi nomi di pacchetti, "com.esempio.platform_learning", "com.esempio.platformLearning".

- Nomi di classi: i nomi delle classi sono scritti seguendo l'annotazione CamelCase, in particolare UpperCamelCase. I nomi delle classi sono in genere nomi o frasi di nomi.
  Ad esempio: "Account" oppure "BankAccount".

- Nomi dei metodi: i nomi dei metodi sono scritti in lowerCamelCase e sono in genere verbi o frasi di verbi.
  Ad esempio, "inviaMessaggio" oppure "ferma".

- Nomi di Interfacce: i nomi delle interfacce sono scritti seguendo l'annotazione CamelCase, in particolare UpperCamelCase. I nomi delle interfacce possono anche essere nomi o frasi di nomi oppure aggettivi o ancora frasi di aggettivo.

- Nomi di classi di test: le classi di test oltre a seguire le linee guida nella sezione "nomi di classi", vengono denominate iniziando con il nome della classe che stanno testando e finendo con la parola "Test". Ad esempio, HashTest.

- Nomi di variabili costanti: i nomi costanti usano la notazione CONSTANT_CASE: tutte le lettere maiuscole, con ogni parola separata dalla successiva da un singolo trattino basso. Ad esempio: PI_GRECO.

- Nomi dei parametri: i nomi dei parametri sono scritti in lowerCamelCase e i parametri formati da un solo carattere nei metodi pubblici dovrebbero essere evitati, questo perché, essendo un metodo pubblico, il nome del parametro può dare informazioni utili nel capire l'utilizzo del metodo.

- Nomi delle variabili locali: i nomi delle variabili locali sono scritti in lowerCamelCase.

- Indentazione:

    1. Non si devono inserire spazi tra il nome del metodo e la parentesi tonda "(" che apre la lista dei parametri.
    2. La parentesi graffa aperta "{" si troverà sulla stessa linea dell'istruzione di dichiarazione.
    3. La parentesi graffa chiusa "}" si troverà su una nuova riga vuota allo stesso livello di indentazione del nome della classe o dell'interfaccia.
    4. Per ogni istruzione all'interno di un blocco usare almeno 4 spazi per l'indentazione.

Esempio:

```
import java.sql.Connection;

public class AnnuncioDAO {

    public synchronized void doSave(OggettoBean annuncio) {
        Connection con = null;
        PreparedStatement ps = null;

        try {
            con = DriverManagerConnectionPool.getConnection();
            ps = con.prepareStatement("INSERT INTO progetto.Oggetto value(?,?,?,?,?,?,?)");

            ps.setString(1, annuncio.getNome());
            ps.setString(2, annuncio.getImmagine());
            ps.setString(3, annuncio.getDescrizione());
            ps.setString(4, annuncio.getCategoria());
            ps.setString(5, annuncio.getRegione());
            ps.setInt(6, annuncio.getPrezzo());
            ps.setString(7, annuncio.getEmail());


            ps.execute();
        } catch (Exception e) {
            e.printStackTrace();
        }finally{

            try {

                ps.close();
                DriverManagerConnectionPool.releaseConnection(con);

            } catch (SQLException e) {

                e.printStackTrace();
            }

        }
    }
```

1.3.3 Pagine JSP (Java servlet page)

Per il codice java contenuto nelle pagine JSP, bisogna rispettare le linee guida della sezione 1.3.2 Classi, interfacce, pacchetti Java.

Per i tag speciali usati nelle JSP:

- Il tag di apertura (<%) è seguito immediatamente dalla fine della riga;
- Il tag di chiusura (%>) si trova all'inizio di una riga;
- È possibile evitare le due regole precedenti, se il corpo del codice Java consiste in una singola istruzione ad esempio: "<%= utente.getNome() %>".

Per il codice HTML ci si deve attenere alla sezione 1.3.4 Pagine HTML.

Esempio:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ page import="model.beans.utenteBean"%>

<!DOCTYPE html>
<html>
<head>
<head>
<meta charset="utf-8">
<title>Registrazione Avvenuta - DiscoverEasy</title>
<link rel="stylesheet"
    href="https://use.fontawesome.com/releases/v5.8.2/css/all.css">
<link rel="stylesheet"
    href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.mi
<link rel="stylesheet"
    href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/css/bootstrap.min.css">
<script
    src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.0/jquery.min.js"></script>
<script
    src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/js/bootstrap.min.js"></script
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="stylesheet/styleRegistrazioneAvvenuta.css">
</head>
</head>
<body>
    <%
        utenteBean utente = (utenteBean) request.getAttribute("datiAnagrafici");
    %>

    <nav class="navbar navbar-default navbar-fixed-top" role="navigation">
        <!-- navbar fissata in alto-->
        <div class="container">

            <div class="navbar-header">
                <!-- Pulsante del menu per finestra piccola -->
                <button type="button" class="navbar-toggle" data-toggle="collapse"
                    data-target="#navbar-collapse-main">
                    <span class="sr-only"> Toggle navigation </span> <span
                        class="icon-bar"></span> <span class="icon-bar"></span> <span
```

1.3.4 Pagine HTML

Usare una sintassi HTML 5 e le seguenti regole di indentazione affinché il codice sia più facile da leggere:

- Un'indentazione consiste in una tabulazione;
- Ogni tag deve avere un'indentazione maggiore del tag che lo contiene;
- Ogni tag di chiusura deve avere lo stesso livello di indentazione del corrispondente tag di apertura;
- I tag di commento devono seguire le stesse regole che si applicano ai tag normali

Utilizzare il protocollo HTTPS per caricare risorse, per immagini e altri file multimediali, fogli di stile e script, a meno che i rispettivi file non siano disponibili su HTTPS.

Ad esempio:

- Non omettere il protocollo

  <script src="//ajax.googleapis.com/ajax/libs/jquery/3.4.0/jquery.min.js"></script>

- Non usare il protocollo http

  <script src="http://ajax.googleapis.com/ajax/libs/jquery/3.4.0/jquery.min.js"></script>

- Usa il protocollo https

  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.0/jquery.min.js"></script>

Per motivi di manutenibilità del codice, mantenere rigorosamente separata la struttura (markup), la presentazione (style), il comportamento (scripting) della pagina e l'interazione tra i tre al minimo assoluto.

Semantica:

Usare gli elementi HTML in base allo scopo per cui sono stati creati. Ad esempio, utilizzare elementi di intestazione per intestazioni, elementi “p” per paragrafi, elementi “a” per ancore, ecc.

Ad esempio:

- Non usare l’elemento “div” per collegamenti ipertestuali

  ```
  <div onclick="goToRecommendations();">All recommendations</div>
  ```

- Usa l’elemento “a” per collegamenti ipertestuali

  ```
  <a href="recommendations/">All recommendations</a>
  ```

L'uso degli elementi HTML in base al suo scopo è importante per motivi di accessibilità, riutilizzo ed efficienza del codice.

Esempio:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Home DiscoverEasy</title>
<link rel="stylesheet"
    href="https://use.fontawesome.com/releases/v5.8.2/css/all.css">
<link rel="stylesheet"
    href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
<link rel="stylesheet"
    href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/css/bootstrap.min.css">
<script
    src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.0/jquery.min.js"></script>
<script
    src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/js/bootstrap.min.js"></script>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="stylesheet/style.css">
</head>
<body>

    <nav class="navbar navbar-default navbar-fixed-top" role="navigation">
        <!-- navbar fissata in alto-->
        <div class="container">

            <div class="navbar-header">
                <!-- Pulsante del menu per finestra piccola -->
                <button type="button" class="navbar-toggle" data-toggle="collapse"
                    data-target="#navbar-collapse-main">
                    <span class="sr-only"> Toggle navigation </span> <span
                        class="icon-bar"></span> <span class="icon-bar"></span> <span
                        class="icon-bar"></span>
                </button>
                <!--Logo-->
                <a href="home.html" id="logo" class="navbar-brand"><img
                    id="immagineLogo" class="img-rounded" src="Img/Logo4.png"
                    alt="logo"></a>
            </div>
```

1.3.5 Pagine CSS

- Indentazione. Indentare gli elementi all'interno del blocco in cui sono contenuti. L'indentazione equivale a una tabulazione. Ad esempio: (Cambia il colore di sfondo dell'elemento <body> in "azzurro" quando la finestra del browser è larga almeno 600 pixel)

```
@media only screen and (max-width: 600px) {
body {
  background-color: lightblue;
  }
}
```

- Fine delle dichiarazioni. Utilizzare un punto e virgola dopo ogni dichiarazione per motivi di estensibilità.

```
/*Modo errato*/
.test {
 display: block;
 height: 100px
}

/* Modo esatto */
.test {
 display: block;
 height: 100px;
}
```

- Spazi. Utilizzare uno spazio dopo i due punti di un nome di proprietà.

Utilizzare sempre uno spazio singolo tra proprietà e valore (ma nessuno spazio tra proprietà e due punti.

```
/* Modo sbagliato */
h3 {
 font-weight:bold;
}

/* Modo esatto */
h3 {
 font-weight: bold;
}
```

- Separazione dei blocchi di dichiarazione. Utilizzare uno spazio tra l'ultimo selettore e il blocco di dichiarazione.

```
 /* Modo errato: mancato spazio */
#video{
  margin-top: 1em;
}

/* Modo errato: inserito parentesi graffa blocco dichiarazione su nuova riga */
#video
{
  margin-top: 1em;
}

/* Modo esatto */
#video {
```

```
  margin-top: 1em;
}
```

- Regola di separazione. Separa i due blocchi di dichiarazione, sempre con due interruzioni di riga.

```
html {
  background: #fff;
}

body {
  margin: auto;
  width: 50%;
}
```

Esempio:

```
#footer {
    color: #ffd700;
    position: fixed;
    bottom: 0;
    background-color: #1e90ff;
    width: 100%;
    height: 100px;
    padding: 3%;
}

.glyphicon {
    color: #ffd700;
}

#contattaci {
    margin-top: 100px;
    padding-top: 75px;
    padding-bottom: 95px;
    border-right: 4px solid #ffd700;
    border-left: 4px solid #ffd700;
}

#bottoneContattaci {
    color: #ffd700;
    background-color: #1e90ff;
    margin-bottom: 30px;
}

a,
a:active,
a:visited,
a:hover {
    color: #ffd700;
    text-decoration: none;
}
```

1.3.6 Pagine JavaScript

Le pagine JavaScript devono seguire le regole per la scrittura di nomi e per quanto riguarda l'indentazione la sezione 1.3.2 Classi, interfacce, pacchetti Java.

1.3.7 SQL

- Keywords di SQL. Le parole chiave (keywords) di SQL vanno scritte in maiuscolo.

- Nomi. I nomi delle tabelle, devono essere scritte in maiuscolo.

- Indentazione

  Usare almeno 4 spazi per indentare un'istruzione all'interno di un blocco.

  La parentesi tonda "(" si troverà sulla linea dell'istruzione di dichiarazione, preceduta da uno spazio.

  La parentesi tonda chiusa ")" si troverà su una nuova riga vuota allo stesso livello di indentazione di dichiarazione del blocco.

Esempio:

```
CREATE DATABASE progetto;

USE progetto;

CREATE TABLE Utente (
    email VARCHAR(150) PRIMARY KEY,
    pass VARCHAR(15) NOT NULL,
    nome_venditore VARCHAR(20) NOT NULL,
    sesso VARCHAR(10),
    anno_nascita INT NOT NULL,
    regione_venditore VARCHAR(25) NOT NULL,
    amministratore INT NOT NULL,
    telefono BIGINT NOT NULL
);

CREATE TABLE Oggetto (
    nome VARCHAR(100) NOT NULL,
    immagine VARCHAR(200) NOT NULL,
    descrizione VARCHAR(2000) NOT NULL,
    categoria VARCHAR(50) NOT NULL,
    regione VARCHAR(25) NOT NULL,
    prezzo INT,
    email_venditore VARCHAR(150) NOT NULL,
    PRIMARY KEY Oggetto (nome , email_venditore),
    FOREIGN KEY (email_venditore)
        REFERENCES Utente (email)
);
```

1.4 Design Pattern

1.4.1 MVC

Il design pattern MVC consente la suddivisione del sistema in tre blocchi principali: Model, View e Controller. Il Model modella i dati del dominio applicativo e fornisce i metodi di accesso ai dati persistenti, il View si occupa della presentazione dei dati all'utente e di ricevere da quest'ultimo gli input, infine il Controller riceve i comandi dell'utente attraverso il View e modifica lo stato di quest'ultimo e del Model.

2. Packages

Il layer presentation è formato dai packages: interfaccia.account, interfaccia.ricerca, Interfaccia.negozio, interfaccia.chat, interfaccia.gestore.

Il layer application è formato dai packages: manager.utente, manager.ricerca, manager.negozio, manager.gestore e manager.chat .

Il layer storage è formato dai packages: model.dao, model.beans.

2.1 Layer Presentation:

2.1.1 interfaccia.account

Il package interfaccia.account è formato da:

- Registrazione.html
- Registrazione.java
- RegisterSucces.jsp
- registrazioneScript.js
- styleRegistrazione.css
- styleRegistrazioneAvvenuta.css

- Login.java
- login.js
- loginRegistrazione.js
- Logout.java
- chisiamo.html
- styleChiSiamo.css
- contattaci.html
- styleContattaci.css
- styleEmailInviata.css
- MioProfilo.jsp
- styleMioProfilo.css
- ModificaDatiMioProfilo.java
- CaricaMioProfilo.java

2.1.2 interfaccia.ricerca

Il package interfaccia.ricerca è formato da:

- Ricerca.java
- PaginaRicerca.jsp
- PaginaRicerca.js
- stylePaginaRicerca.css
- home.html
- style.css
- CaricaPreferiti.java
- AggiungiPreferitiRicerca.java

- RimuoviAnnuncioPreferiti.java
- CestinaAnnunciPreferiti.java
- caricaPreferitiScript.js
- BottoneEliminaOggettoPreferiti.js

2.1.3 interfaccia.negozio

Il package interfaccia.negozio è formato da:

- MioNegozio.jsp
- CaricaAnnunciNegozio.java
- styleMioNegozio.css
- AggiungiAnnuncioNegozio.html
- AggiungiAnnuncioNegozio.java
- styleAggiungiProdotto.css
- oggettoAggiunto.jsp
- styleOggettoAggiunto.css
- EliminaAnnuncioNegozio.java
- AggiungiProdottoScript.js
- EliminaProdottoMioNegozio.js
- inputFile.js
- mioNegozio.js

2.1.4 interfaccia.chat

Il package interfaccia.chat è formato da:

- paginaChat.jsp
- stylePaginaChat.css
- CaricaChat.java
- CaricaMessaggi.java
- IniziaNuovaChat.java
- InviaMessaggio.java
- caricaMessaggi.js
- controllaCasellaMessaggi.js

2.1.5 interfaccia.gestore

Il package interfaccia.gestore è formato da:

- PaginaRicercaGestore.jsp
- stylePaginaRicercaGestore.css
- RicercaGestore.java
- EliminaAccountRicercaGestore.java
- EliminaAnnuncioRicercaGestore.java
- EliminaProdottoAdmin.js
- EliminaAccountAdmin.js
- Page404.html

2.2 Layer Application:

2.2.1 manager.utente

Il package manager.utente è formato da:

- ManagerUtente.java

2.2.2 manager.ricerca

Il package manager.ricerca è formato da:

- ManagerRicerca.java

2.2.3 manager.negozio

Il package manager.negozio è formato da:

- ManagerNegozio.java

2.2.4 manager.chat

Il package manager.chat è formato da:

- ManagerChat.java

2.2.5 manager.gestore

Il package manager.gestore è formato da:

- ManagerGestore.java

2.3 Layer Storage:

2.3.1 model.dao

Il package model.dao è formato da:

- ChatDAO.java
- OggettoDAO.java
- PreferitiDAO.java
- UtenteDAO.java
- DriverManagerConnectionPool.java

2.3.2 model.beans

Il package model.beans è formato da:

- ChatBean.java
- MessaggioBean.java
- OggettoBean.java
- UtenteBean.java

3. Class interfaces

3.1 Package manager.utente :

ManagerUtente.java

La Classe ManagerUtente si occupa della logica applicativa per quanto riguarda l'autenticazione, la registrazione dell'utente al sistema, controllando il formato dei campi immessi dall'utente, se è un utente bannato o eliminato, il caricamento dei dati personali dell'utente, le operazioni sui di esse come modifica password ed infine dell'eliminazione dell'Account.

Metodo	public UtenteBean autentica (String email, String password)
Descrizione	Il metodo controlla se i parametri presi in input String email e String password, corrispondono alle credenziali un utente presente nel database.
Context	ManagerUtente::autentica (email :String, password :String)
Precondizione	email != null AND password != null.
Post condizione	ritorna un oggetto UtenteBean contenente le informazioni dell'utente nel database con attributo "email_ban" uguale null e con gli attributi "email" e "password" uguali rispettivamente alle stringhe email e password presi in input altrimenti ritorna null.

Metodo	UtenteBean caricaMioProfilo (String email)
Descrizione	Il metodo ritorna i dati personali di un utente presente sul database, con email uguale al parametro preso in input String email.
Context	ManagerUtente::caricaMioProfilo(email :String)
Precondizione	Sul database esiste un utente con email uguale alla stringa email presa in input.
Post condizione	Ritorna un oggetto UtenteBean con le informazioni di un utente presente sul database con email uguale alla stringa email presa in input.

Metodo	public void eliminaAccount(String email) throws SQLException
Descrizione	Il metodo elimina l'utente sul database con attributo "email" uguale alla stringa presa in input email, e gli annunci che ha salvato ai preferiti e pubblicati.
Context	ManagerUtente::modificaPassword (email :String, password :String)
Precondizione	Sul database esiste un utente con attributo "email" uguale alla stringa email presa in input e per la stringa password presa in input deve essere vera la seguente espressione regolare: ^[0-9a-z]{8,40}$
Post condizione	L'attributo "password" dell'utente sul database con attributo email uguale alla stringa email presa in input è uguale alla stringa password presa in input.

Metodo	public UtenteBean checkExistent(String email)
Descrizione	Il metodo verifica se esiste o meno sul database un utente con email uguale alla stringa presa in input email.
Context	ManagerUtente::checkExistent (email :String)
Precondizione	email != null .
Post condizione	Il metodo ritorna un oggetto UtenteBean con le informazioni dell'utente sul database, se esiste, con attributo email uguale alla stringa email presa in input, altrimenti ritorna null.

Metodo	public UtenteBean registra(String email, String password, String nome, String sesso, String regione, String telefono, String annoNascita, String condizioniUtente) throws IllegalArgumentException, SQLException
Descrizione	Il metodo registra un utente sul database con i parametri presi in input.

Context	ManagerUtente::registra (email :String, password :String, nome :String, sesso :String, regione :String, telefono :String, annoNascita :String, condizioniUtente :String)
Precondizione	Non esiste sul database un utente con email uguale alla stringa email presa in input.
Precondizione	Per la stringa presa in input email deve essere vera la seguente espressione regolare, ^[a-zA-Z0-9_+&*-]+(?:\\.[a-zA-Z0-9_+&*-]+)*@(?:[a-zA-Z0-9-]+\\.)+[a-zA-Z]{2,7}$.
Precondizione	Per la stringa presa in input password deve essere vera la seguente espressione regolare, ^[0-9a-z]{8,40}$.
Precondizione	Per la stringa presa in input nome deve essere vera la seguente espressione regolare, ^[a-zA-Z]{1, 30}$.
Precondizione	sesso. equalsIgnoreCase (“maschio”) = true OR sesso.equalsIgnoreCase(“femmina”) = true .
Precondizione	regione.equals("Campania") = true OR regione.equals("Abruzzo") = true OR regione.equals("Basilicata") = true OR regione.equals("Calabria") = true OR regione.equals("Emilia-Romagna") = true OR regione.equals("Friuli-Venezia-Giulia") = true OR regione.equals("Lazio") = true OR regione.equals("Liguria") = true OR regione.equals("Lombardia") = true OR regione.equals("Marchie") = true OR regione.equals("Molise") = true OR regione.equals("Piemonte") = true OR regione.equals("Puglia") = true OR regione.equals("Sardegna") = true OR regione.equals("Sicilia") = true OR regione.equals("Toscana") = true OR regione.equals("Trentino-Alto-Adige") = true OR regione.equals("Umbria") = true OR regione.equals("Valle d'Aosta") = true .
Precondizione	Per la stringa presa in input telefono deve essere vera la seguente espressione regolare, ^[0-9]{9,10}$:
Precondizione	Integer.parseInt(annoNascita) > 1918 AND Integer.parseInt(annoNascita) < 2003 .
Precondizione	(condizioniUtente).equals(“on”) .
Precondizione	Ritorna un oggetto UtenteBean con le informazioni di un utente presente sul database con attributi email, pass, nome_venditore, sesso, anno_nascita, regione_venditore, telefono uguali rispettivamente alle stringhe prese in input email, password, nome, sesso, annoNascita, regione, telefono .

3.2 Package manager.ricerca

ManagerRicerca.java

La classe ManagerRicerca si occupa della logica applicativa per quanto riguarda la ricerca, e la gestione dei preferiti, come l'aggiunta o l'eliminazione di un annuncio ai preferiti, e la consistenza dei preferiti nel database.

Metodo	public ArrayList<OggettoBean> caricaAnnunci(String email) throws SQLException
Descrizione	Il metodo restituisce la lista di annunci che ha pubblicato l'utente che ha l'email uguale alla stringa email presa in input.
Context	ManagerRicerca::caricaAnnunci (email :String)
Precondizione	Email è presente sul database.
Post condizione	Il metodo restituisce tutti gli annunci sul database che hanno l'attributo "eliminato" uguale a false, l'attributo "email_ban" uguale a null e che hanno l'attributo "email" uguale alla stringa email presa in input.

Metodo	public void cestinaAnnunci(String emailUtente) throws SQLException
Descrizione	Il metodo elimina tutti i preferiti dell'utente sul database che ha l'attributo "email" uguale alla stringa email presa in input.
Context	ManagerRicerca::cestinaAnnunci (emailUtente:String)
Precondizione	Nel database è presente un utente con attributo "email" uguale alla stringa in input emailUtente.

Post condizione	Nel database non sono presenti preferiti con attributo “email” uguale alla stringa emailUtente presa in input.

Metodo	public void eliminaAnnuncio(String emailUtente, String emailVenditore, String nomeAnnuncio, GregorianCalendar dataOra) throws SQLException
Descrizione	Il metodo elimina l’annuncio con i parametri presi in input dal database.
Context	ManagerRicerca::eliminaAnnuncio (emailUtente:String, emailVenditore :String, nomeAnnuncio :String, dataOra :GregorianCalendar)
Precondizione	Esiste sul database un annuncio ai preferiti con attributi email_utente, email_venditore, nome_annuncio uguali rispettivamente ai parametri presi in input, emailUtente, emailVenditore, nomeAnnuncio, e con un attributo “data_ora_annuncio “ uguale al seguente formato di concatenazioni stringhe java, dataOra.get(GregorianCalendar.YEAR)+"-"+ dataOra.get(GregorianCalendar.MONTH)+1+"-"+dataOra.get(GregorianCalendar.DAY_OF_MONTH)+" "+dataOra.get(GregorianCalendar.HOUR_OF_DAY)+":"+ dataOra.get(GregorianCalendar.MINUTE)+":"+ dataOra.get(GregorianCalendar.SECOND),
Post condizione	L’insieme dei preferiti sul database dell’utente con attributo “email” uguale alla stringa emailUtente presa in input prima dell’esecuzione del metodo, è uguale all’insieme dei preferiti sul database dell’utente con attributo “email” uguale alla stringa emailUtente presa in input dopo l’esecuzione del metodo, senza l’annuncio preferito con attributi email_utente, email_venditore, nome_annuncio uguali rispettivamente ai parametri presi in input, emailUtente, emailVenditore, nomeAnnuncio, e con un attributo “data_ora_annuncio “ uguale al seguente formato di concatenazioni stringhe java, dataOra.get(GregorianCalendar.YEAR)+"-"+ dataOra.get(GregorianCalendar.MONTH)+1+"-"+dataOra.get(GregorianCalendar.DAY_OF_MONTH)+" "+dataOra.get(GregorianCalendar.HOUR_OF_DAY)+":"+ dataOra.get(GregorianCalendar.MINUTE)+":"+ dataOra.get(GregorianCalendar.SECOND

Metodo	public Hashtable<String, OggettoBean> mergePreferitiSessioneDatabase(Hashtable<String, OggettoBean> insiemeSessione, String email_utente) throws SQLException
Descrizione	Il metodo unisce l’insieme di annunci preso in input, insiemeSessione con l’insieme dei preferiti presenti sul database dell’utente con attributo “email” uguale alla stringa emai_utente presa in input.
Context	ManagerRicerca::mergePreferitiSessioneDatabase (insiemeSessione :Hashtable<String, OggettoBean>, email_utente :String)
Precondizione	Nel database è presente un utente con attributo “email” uguale alla stringa email_utente presa in input.
Precondizione	insiemeSessione != null.
Post condizione	L’insieme ritornato è uguale all’insieme preferiti dell’utente presente sul database con attributo “email” uguale alla stringa email presa in input.

Post condizione	L'insieme preferiti dell'utente presente sul database con attributo "email" uguale alla stringa email presa in input, è uguale al suo insieme preferiti prima che venisse eseguito il metodo, unito l'insieme preso in input insiemeSessione.

Metodo	public ArrayList<OggettoBean> ricercaAnnunci(String regione, String categoria, String barraRicerca)
Descrizione	Il metodo ricerca gli annunci sul database e li restituisce, i cui attributi rispettano i parametri presi in input.
Context	ManagerRicerca::ricercaAnnunci (regione :String, categoria :String, barraRicerca :String)
Precondizione	regione != null AND categoria != null AND barraRicerca != null.
Post condizione	Restituisce sottoforma di insieme tutti gli annunci sul database con attributi "regione", "categoria" uguali rispettivamente ai parametri in input regione, categoria, e con attributo "nome" che contiene il parametro in input barraRicerca.

Metodo	public OggettoBean ricercaAnnuncioPerChiave (String nomeAnnuncio, String emailVenditore, GregorianCalendar dataOraAnnuncio) throws SQLException
Descrizione	Il metodo restituisce l'annuncio del database se presente che ha gli attributi pertinenti ai parametri in input altrimenti ritorna null.
Context	ManagerRicerca::ricercaAnnuncioPerChiave (nomeAnnuncio :String, emailVenditore :String, dataOraAnnuncio :GregorianCalendar)
Precondizione	nomeAnnuncio != null AND emailVenditore != null AND dataOraAnnuncio != null.
Post condizione	Restituisce dal database l'annuncio , se presente, che ha gli attributi "nome", "email_venditore" uguali rispettivamente ai parametri presi in input nomeAnnuncio, emailVenditore, e con un attributo "data_ora_annuncio " uguale al seguente formato di concatenazioni stringhe java, dataOraAnnuncio.get(GregorianCalendar.YEAR)+"-"+ dataOraAnnuncio.get(GregorianCalendar.MONTH)+1+"-"+ dataOraAnnuncio.get(GregorianCalendar.DAY_OF_MONTH)+" "+ dataOraAnnuncio.get(GregorianCalendar.HOUR_OF_DAY)+":"+ dataOraAnnuncio.get(GregorianCalendar.MINUTE)+":"+ dataOraAnnuncio.get(GregorianCalendar.SECOND), altrimenti restituisce null.

Metodo	public void salvaAnnuncioAiPreferiti(String emailUtente, OggettoBean annuncio) throws SQLException
Descrizione	Il metodo salva sul database l'annuncio preso in input all'utente con attributo "email" uguale al parametro in input emailUtente
Context	ManagerRicerca::salvaAnnuncioAiPreferiti (emailUtente :String, annuncio :OggettoBean)

Precondizione	sul database non è presente un preferito con attributi “email_utente”, “email_venditore”, “nome_annuncio”, uguali rispettivamente ai parametri in input emailUtente, annuncio.getEmail(), annuncio.getNome(), e con un attributo “data_ora_annuncio “ uguale al seguente formato di concatenazioni stringhe java, annuncio.getDataOra().get(GregorianCalendar.YEAR)+"-"+ annuncio.getDataOra().get(GregorianCalendar.MONTH)+1+"-"+ annuncio.getDataOra().get(GregorianCalendar.DAY_OF_MONTH)+" "+ annuncio.getDataOra().get(GregorianCalendar.HOUR_OF_DAY)+":"+ annuncio.getDataOra().get(GregorianCalendar.MINUTE)+":"+ annuncio.getDataOra().get(GregorianCalendar.SECOND).
Precondizione	sul database è presente un annuncio con attributi “nome”, “immagine”, “descrizione”, “categoria”, “regione”, “prezzo”, “email_venditore” sono uguali rispettivamente ai parametri in input annuncio.getNome(), annuncio.getImmagine(), annuncio.getDescrizione(), annuncio.getCategoria(), annuncio.getRegione(), annuncio.getPrezzo(), emailUtente, e con un attributo “data_ora_annuncio “ uguale al seguente formato di concatenazioni stringhe java, annuncio.getDataOra().get(GregorianCalendar.YEAR)+"-"+ annuncio.getDataOra().get(GregorianCalendar.MONTH)+1+"-"+ annuncio.getDataOra().get(GregorianCalendar.DAY_OF_MONTH)+" "+ annuncio.getDataOra().get(GregorianCalendar.HOUR_OF_DAY)+":"+ annuncio.getDataOra().get(GregorianCalendar.MINUTE)+":"+ annuncio.getDataOra().get(GregorianCalendar.SECOND).
Post condizione	All’insieme dei preferiti sul database prima di eseguire il metodo, dell’utente con attributo “email” uguale al parametro in input email_utente, è aggiunto un preferito con attributi “email_utente”, “email_venditore”, “nome_annuncio”, uguali rispettivamente ai parametri in input emailUtente, annuncio.getEmail(), annuncio.getNome(), e con un attributo “data_ora_annuncio “ uguale al seguente formato di concatenazioni stringhe java, annuncio.getDataOra().get(GregorianCalendar.YEAR)+"-"+ annuncio.getDataOra().get(GregorianCalendar.MONTH)+1+"-"+ annuncio.getDataOra().get(GregorianCalendar.DAY_OF_MONTH)+" "+ annuncio.getDataOra().get(GregorianCalendar.HOUR_OF_DAY)+":"+ annuncio.getDataOra().get(GregorianCalendar.MINUTE)+":"+ annuncio.getDataOra().get(GregorianCalendar.SECOND).

3.3 package manager.negozio

ManagerNegozio.java

La classe ManagerNegozio si occupa della logica applicativa del salvataggio di un annuncio nel negozio di utente, verificando che rispetti criteri di formato, e dell'eliminazione di un annuncio pubblicato da un utente.

Metodo	public void salvaAnnuncio(String email, String nome, String categoria, String regione, String descrizione, String prezzo, String urlImmagine) throws SQLException
Descrizione	Il metodo salva l'annuncio con i parametri presi in input all'utente sul database con attributo "email" uguale al parametro in input email.
Context	ManagerNegozio:: salvaAnnuncio (email :String, nome :String, categoria :String, regione :String, descrizione :String, prezzo :String, urlImmagine :String)
Precondizione	email != null AND nome != null AND categoria != null AND regione != null AND descrizione != null AND prezzo != null AND urlImmagine != null.
Post condizione	All'insieme di annunci sul database dell'utente con attributo "email" uguale al parametro in input email, prima di eseguire il metodo, è aggiunto un annuncio con attributi "nome", "immagine", "descrizione", "categoria", "regione", "prezzo", "email_venditore" uguali rispettivamente ai parametri in input nome, urlImmagine, descrizione, categoria, regione, prezzo, email.

Metodo	public void eliminaAnnuncio(String nome, String email, GregorianCalendar data_ora) throws SQLException
Descrizione	Il metodo elimina l'annuncio sul database con chiave primaria uguale ai parametri in input.
Context	ManagerNegozio::eliminaAnnuncio (nome :String, email :String, data_ora :GregorianCalendar)
Precondizione	Sul database è presente un annuncio con attributi "nome", "email_venditore", uguali rispettivamente ai parametri in input nome, email e con attributo "data_ora" uguale al seguente formato di concatenazioni stringhe java, data_ora.get(GregorianCalendar.YEAR)+"-"+ data_ora.get(GregorianCalendar.MONTH)+1+"-"+ data_ora.get(GregorianCalendar.DAY_OF_MONTH)+" "+ data_ora.get(GregorianCalendar.HOUR_OF_DAY)+":"+ data_ora.get(GregorianCalendar.MINUTE)+":"+ data_ora.get(GregorianCalendar.SECOND).
Post condizione	L'insieme di annunci sul database dell'utente con attributo "email" uguale al parametro in input email, è uguale all'insieme di annunci dell'utente prima di eseguire il metodo senza l'annuncio con attributi "nome", "email_venditore", uguali rispettivamente ai parametri in input nome, email e con attributo "data_ora" uguale al seguente formato di concatenazioni stringhe java, data_ora.get(GregorianCalendar.YEAR)+"-"+ data_ora.get(GregorianCalendar.MONTH)+1+"-"+ data_ora.get(GregorianCalendar.DAY_OF_MONTH)+" "+ data_ora.get(GregorianCalendar.HOUR_OF_DAY)+":"+ data_ora.get(GregorianCalendar.MINUTE)+":"+ data_ora.get(GregorianCalendar.SECOND).

3.4 Package manager.gestore

ManagerGestore.java

La classe ManagerGestore si occupa della logica applicativa delle funzioni per il ban dell'annuncio e dell'account.

Metodo	public void banAccount(String emailGestore, String emailBan) throws SQLException
Descrizione	Il metodo banna l'account dell'utente sul database con attributo "email" uguale alla stringa in input emailBan, eliminando tutti i suoi preferiti e i suoi annunci pubblicati.
Context	ManagerGestore::banAccount (emailGestore :String, emailBan:String)
Precondizione	Esiste sul database un utente con attributo "email" uguale al parametro in input emailBan.
Precondizione	Esiste sul database un utente con attributo "email" uguale al parametro in input emailGestore.
Post condizione	L'utente sul database con attributo "email" uguale al parametro in input emailBan, ha un attributo "email_ban" uguale alla stringa presa in input emailGestore.
Post condizione	Gli annunci sul database con attributo "email_venditore" uguale al parametro in input emailBan, hanno un attributo "email_ban" uguale al parametro in input emailGestore.
Post condizione	I preferiti sul database con attributo "email_utente" uguale al parametro in input emailBan, hanno l'attributo eliminato uguale a true.

Metodo	public void banAnnuncio(String email_gestore, String nome, String email_annuncio, GregorianCalendar data_ora) throws SQLException
Descrizione	Il metodo banna l'annuncio sul database pertinente ai parametri presi in input nome, email_annuncio, data_ora.
Context	ManagerGestore::banAnnuncio (email_gestore :String, nome :String, email_annuncio :String, data_ora :GregorianCalendar)
Precondizione	Esiste un annuncio sul database con attributi "nome", "email_venditore", uguali rispettivamente ai parametri in input nome, email_annuncio e con attributo "data_ora" uguale al seguente formato di concatenazioni stringhe java, data_ora.get(GregorianCalendar.YEAR)+"-"+ data_ora.get(GregorianCalendar.MONTH)+1+"-"+

	data_ora.get(GregorianCalendar.DAY_OF_MONTH)+" "+ data_ora.get(GregorianCalendar.HOUR_OF_DAY)+":"+ data_ora.get(GregorianCalendar.MINUTE)+":"+ data_ora.get(GregorianCalendar.SECOND).
Precondizione	Esiste sul database un utente con attributo “email” uguale al parametro in input email_gestore.
Post condizione	L’ annuncio sul database con attributi “nome”, “email_venditore”, uguali rispettivamente ai parametri in input nome, email_annuncio e con attributo “data_ora” uguale al seguente formato di concatenazioni stringhe java, data_ora.get(GregorianCalendar.YEAR)+"-"+ data_ora.get(GregorianCalendar.MONTH)+1+"-"+ data_ora.get(GregorianCalendar.DAY_OF_MONTH)+" "+ data_ora.get(GregorianCalendar.HOUR_OF_DAY)+":"+ data_ora.get(GregorianCalendar.MINUTE)+":"+ data_ora.get(GregorianCalendar.SECOND), ha l’attributo email_ban uguale al parametro in input email_gestore.

3.5 Package manager.chat

ManagerChat.java

La classe ManagerChat si occupa della logica applicativa per quanto riguarda la chat, come la gestione dell'invio dei messaggi, il caricamento della chat e dei messaggi.

Metodo	public ArrayList<ChatBean> caricaChat(String emailUtente) throws SQLException
Descrizione	Il metodo carica le chat dal database dell'utente con attributo "email" uguale al parametro in input emailUtente.
Context	ManagerChat::caricaChat (email :String)
Precondizione	Sul database è presente un utente con attributo "email" uguale al parametro in input email.
Post condizione	L'insieme restituito dal metodo è l'insieme delle chat sul database dell'utente con attributo "email" uguale al parametro in input email.

Metodo	public ArrayList<MessaggioBean> caricaMessaggi(String emailUtente, String emailMittenteChat, String emailDestinatarioChat, String nomeAnnuncio, GregorianCalendar dataOraAnnuncio) throws SQLException
Descrizione	Il metodo restituisce l'insieme dei messaggi di una determinata chat di un utente dal database pertinente ai parametri in input.
Context	ManagerChat::caricaMessaggi(emailUtente :String, emailMittenteChat :String, emailDestinatarioChat :String, nomeAnnuncio :String, dataOraAnnuncio :GregorianCalendar)
Precondizione	Sul database è presente un utente con attributo "email" uguale al parametro in input emailUtente.
Precondizione	Sul database è presente una chat con attributi "email_mittente", "email_destinatario", "nome_annuncio" uguali rispettivamente ai parametri in input emailMittenteChat, emailDestinatarioChat, nomeAnnuncio e con attributo "data_ora_annuncio" uguale al seguente formato di concatenazioni stringhe java, dataOraAnnuncio.get(GregorianCalendar.YEAR)+"-"+ dataOraAnnuncio.get(GregorianCalendar.MONTH)+1+"-"+ dataOraAnnuncio.get(GregorianCalendar.DAY_OF_MONTH)+" "+ dataOraAnnuncio.get(GregorianCalendar.HOUR_OF_DAY)+":"+

	dataOraAnnuncio.get(GregorianCalendar.MINUTE)+":"+ dataOraAnnuncio.get(GregorianCalendar.SECOND).
Post condizione	Restituisce l'insieme dei messaggi sul database con attributi "email_mittente_chat", "email_destinatario_chat", "nome_annuncio", uguali ai parametri in input emailMittenteChat, emailDestinatarioChat, nomeAnnuncio, e con attributo "data_ora_annuncio" uguale al seguente formato di concatenazioni stringhe java, dataOraAnnuncio.get(GregorianCalendar.YEAR)+"-"+ dataOraAnnuncio.get(GregorianCalendar.MONTH)+1+"-"+ dataOraAnnuncio.get(GregorianCalendar.DAY_OF_MONTH)+" "+ dataOraAnnuncio.get(GregorianCalendar.HOUR_OF_DAY)+":"+ dataOraAnnuncio.get(GregorianCalendar.MINUTE)+":"+ dataOraAnnuncio.get(GregorianCalendar.SECOND).
Post condizione	Se il parametro in input emailUtente è uguale al parametro in input emailMittenteChat, allora l'attributo "mittente_messaggio_non_letto" sul database della chat in precondizione=0, altrimenti l'attributo "destinatario_messaggio_non_letto" è uguale a 0.

Metodo	public void iniziaNuovaChat(String emailMittenteChat, String emailDestinatarioChat, String nomeAnnuncio, GregorianCalendar dataOraAnnuncio) throws SQLException
Descrizione	Il metodo inizia una nuova chat tra un utente mittente e l'utente che ha pubblicato un annuncio.
Context	ManagerChat::iniziaNuovaChat(emailMittenteChat :String, emailDestinatarioChat :String, nomeAnnuncio :String, dataOraAnnuncio :GregorianCalendar)
Precondizione	: Sul database è presente un utente con attributo "email" uguale al parametro in input emailMittenteChat.
Precondizione	Sul database è presente un utente con attributo "email" uguale al parametro in input emailDestinatarioChat.
Precondizione	Esiste un annuncio sul database con attributi "nome", "email_venditore", uguali rispettivamente ai parametri in input nome, emailDestinatarioChat e con attributo "data_ora_annuncio" uguale al seguente formato di concatenazioni stringhe java, dataOraAnnuncio.get(GregorianCalendar.YEAR)+"-"+ dataOraAnnuncio.get(GregorianCalendar.MONTH)+1+"-"+ dataOraAnnuncio.get(GregorianCalendar.DAY_OF_MONTH)+" "+ dataOraAnnuncio.get(GregorianCalendar.HOUR_OF_DAY)+":"+ dataOraAnnuncio.get(GregorianCalendar.MINUTE)+":"+ dataOraAnnuncio.get(GregorianCalendar.SECOND).
Precondizione	Sul database non è presente una chat con attributi "email_mittente", "email_destinatario", "nome_annuncio" uguali rispettivamente ai parametri in input emailMittenteChat, emailDestinatarioChat, nomeAnnuncio e con attributo "data_ora_annuncio" uguale al seguente formato di concatenazioni stringhe java, dataOraAnnuncio.get(GregorianCalendar.YEAR)+"-"+ dataOraAnnuncio.get(GregorianCalendar.MONTH)+1+"-"+ dataOraAnnuncio.get(GregorianCalendar.DAY_OF_MONTH)+" "+ dataOraAnnuncio.get(GregorianCalendar.HOUR_OF_DAY)+":"+ dataOraAnnuncio.get(GregorianCalendar.MINUTE)+":"+ dataOraAnnuncio.get(GregorianCalendar.SECOND).
Post condizione	Sul database è presente una chat con attributi "email_mittente", "email_destinatario", "nome_annuncio" uguali rispettivamente ai parametri in input emailMittenteChat,

	emailDestinatarioChat, nomeAnnuncio e con attributo “data_ora_annuncio” uguale al seguente formato di concatenazioni stringhe java, dataOraAnnuncio.get(GregorianCalendar.YEAR)+"-"+ dataOraAnnuncio.get(GregorianCalendar.MONTH)+1+"-"+ dataOraAnnuncio.get(GregorianCalendar.DAY_OF_MONTH)+" "+ dataOraAnnuncio.get(GregorianCalendar.HOUR_OF_DAY)+":"+ dataOraAnnuncio.get(GregorianCalendar.MINUTE)+":"+ dataOraAnnuncio.get(GregorianCalendar.SECOND).

Metodo	public void inviaMessaggio(String descrizione, String emailMittenteMessaggio, String emailMittenteChat, String emailDestinatarioChat, String nomeAnnuncio, GregorianCalendar dataOraAnnuncio) throws SQLException
Descrizione	Il metodo invia un messaggio con descrizione uguale al parametro in input descrizione, alla chat corrispondente ai parametri in input emailMittenteChat, emailDestinatarioChat, nomeAnnuncio, dataOraAnnuncio.
Context	ManagerChat::inviaMessaggio(descrizione :String, emailMittenteMessaggio :String, emailMittenteChat :String, emailDestinatarioChat :String, nomeAnnuncio :String, dataOraAnnuncio :GregorianCalendar)
Precondizione	descrizione != null.
Precondizione	Sul database è presente un utente con attributo “email” uguale al parametro in input emailMittenteChat.
Precondizione	Sul database è presente un utente con attributo “email” uguale al parametro in input emailDestinatarioChat.
Precondizione	Esiste un annuncio sul database con attributi “nome”, “email_venditore”, uguali rispettivamente ai parametri in input nome, emailDestinatarioChat e con attributo “data_ora_annuncio” uguale al seguente formato di concatenazioni stringhe java, dataOraAnnuncio.get(GregorianCalendar.YEAR)+"-"+ dataOraAnnuncio.get(GregorianCalendar.MONTH)+1+"-"+ dataOraAnnuncio.get(GregorianCalendar.DAY_OF_MONTH)+" "+ dataOraAnnuncio.get(GregorianCalendar.HOUR_OF_DAY)+":"+ dataOraAnnuncio.get(GregorianCalendar.MINUTE)+":"+ dataOraAnnuncio.get(GregorianCalendar.SECOND).
Precondizione	Sul database è presente una chat con attributi “email_mittente”, “email_destinatario”, “nome_annuncio” uguali rispettivamente ai parametri in input emailMittenteChat, emailDestinatarioChat, nomeAnnuncio e con attributo “data_ora_annuncio” uguale al seguente formato di concatenazioni stringhe java, dataOraAnnuncio.get(GregorianCalendar.YEAR)+"-"+ dataOraAnnuncio.get(GregorianCalendar.MONTH)+1+"-"+ dataOraAnnuncio.get(GregorianCalendar.DAY_OF_MONTH)+" "+ dataOraAnnuncio.get(GregorianCalendar.HOUR_OF_DAY)+":"+ dataOraAnnuncio.get(GregorianCalendar.MINUTE)+":"+ dataOraAnnuncio.get(GregorianCalendar.SECOND).
Post condizione	Salva sul database un messaggio con attributi “email_mittente_chat”, “email_destinatario_chat”, “nome_annuncio”, “email_mittente_messaggio”, “descrizione” uguali ai parametri in input emailMittenteChat, emailDestinatarioChat, nomeAnnuncio, emailMittenteMessaggio, descrizione e con attributo “data_ora_annuncio” uguale al seguente formato di concatenazioni stringhe java, dataOraAnnuncio.get(GregorianCalendar.YEAR)+"-"+

	dataOraAnnuncio.get(GregorianCalendar.MONTH)+1+"-"+ dataOraAnnuncio.get(GregorianCalendar.DAY_OF_MONTH)+" "+ dataOraAnnuncio.get(GregorianCalendar.HOUR_OF_DAY)+":"+ dataOraAnnuncio.get(GregorianCalendar.MINUTE)+":"+ dataOraAnnuncio.get(GregorianCalendar.SECOND).
Post condizione	Se il parametro in input emailMittenteMessaggio è uguale al parametro in input emailMittenteChat, allora l'attributo "mittente_messaggio_non_letto" sul database della chat in precondizione = 0, altrimenti l'attributo "destinatario_messaggio_non_letto" = 0.

Metodo	public boolean notifica (String emailUtente) throws SQLException
Descrizione	Il metodo restituisce true se l'utente sul database con attributo "email" che è uguale al parametro in input emailUtente, ha un messaggio non letto, altrimenti false.
Context	ManagerChat::notifica (emailUtente :String)
Precondizione	Sul database esiste un utente con attributo "email" uguale al parametro in input emailUtente.
Post condizione	Il metodo restituisce true se esiste sul database almeno una chat con attributo "email_mittente" o attributo "email_destinatario" uguale al parametro in input emailUtente, e con attributo "mittente_messaggio_non_letto" o "destinatario_messaggio_non_letto" uguale a 1.

3.6 Package model.dao

UtenteDAO.java

La classe UtenteDAO si interfaccia sul database per eseguire operazioni di inserimento di un utente, che serve per la registrazione di un nuovo account; operazioni di eliminazione di un utente che può eseguire il possessore di un account per eliminare il proprio; operazione di ban di un utente eseguita dal gestore, e l'operazione per recuperare le informazioni di un utente, usate per effettuare il login e per caricare i dati dell'utente.

Metodo	public UtenteBean doRetriveByKey(String email);
Descrizione	Il metodo restituisce le informazioni dal database dell'utente pertinente ai parametri di ricerca se esiste, altrimenti null.
Context	Context UtenteDAO::doRetriveByKey(email :String)
Precondizione	email != null.
Post condizione	Il metodo ritorna le informazioni dal database , nell'oggetto UtenteBean dell'utente con attributo "email" uguale al parametro in input email, se esiste, altrimenti ritorna null.

Metodo	public void doSave(UtenteBean utente);
Descrizione	Il metodo salva le informazioni di un utente, contenute nell'oggetto in input utente sul database.
Context	UtenteDAO::doSave(utente :UtenteBean)
Precondizione	utente != null.
Precondizione	Sul database non esiste un utente con attributo "email" uguale al parametro in input utente.getEmail();
Post condizione	Sul database esiste un utente con attributi "email", "pass", "nome_venditore", "sesso", "anno_nascita", "regione_venditore", "amministratore", "telefono", "eliminato", "email_ban", uguali rispettivamente ad utente.getEmail(), utente.getPass(), utente.getNome(), utente.getSesso(), utente.getAnno_nascita(), utente.getRegione(), 0, utente.getTelefono(), false, null.

Metodo	public synchronized void removeByKey(String email) throws SQLException
Descrizione	Il metodo elimina l'utente dal database con attributo "email" uguale al parametro in input email.
Context	UtenteDAO::removeByKey (email :UtenteBean)
Precondizione	Sul database esiste un utente con attributo "email" uguale al parametro in input email.
Post condizione	Sul database non esiste un utente con attributo "email" uguale al parametro in input email.

Metodo	public synchronized void doChangePassword(String email, String password) throws SQLException
Descrizione	Il metodo modifica la password all'utente con attributo "email" sul database uguale al parametro in input email.
Context	UtenteDAO:: doChangePassword (email :String, password :String)
Precondizione	password != null.
Precondizione	Sul database esiste un utente con attributo "email" uguale al parametro in input email.
Post condizione	L'attributo "pass" sul database, dell'utente con attributo "email" uguale al parametro in input email, è uguale al parametro in input password.

Metodo	public synchronized void deleteByKey(String email) throws SQLException
Descrizione	Il metodo setta a true l'attributo "elimina" all'utente sul database con "email" uguale al parametro in input email.
Context	UtenteDAO:: deleteByKey (email :String)
Precondizione	Sul database esiste un utente con attributo "email" uguale al parametro in input email e con attributo "email_ban" uguale a null.
Post condizione	L'attributo "elimina", dell'utente sul database con attributo "email" uguale al parametro in input email, e con attributo "eliminato" uguale a true e con attributo "email_ban" uguale a null.

Metodo	public synchronized void banByKey(String emailGestore, String emailBan) throws SQLException
Descrizione	Il metodo banna l'utente sul database con attributo "email" uguale al parametro in input emailBan.
Context	UtenteDAO:: banByKey (emailGestore :String, emailBan :String)
Precondizione	Sul database esiste un utente con attributo "email" uguale al parametro in input emailBan e con attributo "eliminato" = false.
Precondizione	Sul database esite un utente con attributo "email" uguale al parametro in input emailGestore.

Post condizione	L'attributo "email_ban" sul database, dell'utente con attributo "email" uguale al parametro in input emailBan, è uguale al parametro in input emailGestore, e l'attributo "eliminato" è uguale a false.

PreferitiDAO.java

La classe si interfaccia sul database per eseguire le operazioni di salvataggio, eliminazione e ricerca dei preferiti sul database.

Metodo	public synchronized Hashtable<String, OggettoBean> doSearchByEmail(String email_utente) throws SQLException
Descrizione	Il metodo recupera e ritorna attraverso un insieme tutti gli annunci aggiunti ai preferiti che non sono stati bannati o eliminati dell'utente con attributo sul database "email" uguale al parametro in input email_utente.
Context	UtenteDAO::doSearchByEmail (email_utente :String)
Precondizione	Sul database esiste un utente con attributo "email" uguale al parametro in input email_utente, con attributi "email_ban" ed "eliminato" uguali rispetivamente ai valori null, false
Post condizione	Ritorna l'insieme dei preferiti sul database con attributo "eliminato" uguale false, dell'utente con attributo "email" uguale al parametro in input email_utente.
Post condizione	L'insieme di preferiti ritornato sono annunci sul databse che hanno attributo "eliminato" uguale a false, e "email_ban" uguale a null.

Metodo	public synchronized void doSave(String emailUtente, String emailVenditore, String nomeAnnuncio, GregorianCalendar dataOra) throws SQLException
Descrizione	Il metodo salva le chiavi primarie prese in input dell'annuncio, ai preferiti sul database dell'utente con attributo "email" uguale al parametro in input emailUtente.
Context	UtenteDAO:: doSave (emailUtente :String, emailVenditore :String, nomeAnnuncio :String, dataOra :GregorianCalendar)
Precondizione	sul database esiste un utente con attributo "email" uguale al parametro in input emailUtente.
	esiste un annuncio sul database con attributi "nome", "email_venditore", uguali rispettivamente ai parametri in input nomeAnnuncio, emailVenditore e con attributo "data_ora_annuncio" uguale al seguente formato di concatenazioni stringhe java, dataOra.get(GregorianCalendar.YEAR)+"-"+ dataOra.get(GregorianCalendar.MONTH)+1+"-"+ dataOra.get(GregorianCalendar.DAY_OF_MONTH)+" "+

	dataOra.get(GregorianCalendar.HOUR_OF_DAY)+":"+ dataOra.get(GregorianCalendar.MINUTE)+":"+ dataOra.get(GregorianCalendar.SECOND).
Post condizione	Esiste sul database un annuncio ai preferiti con attributi “email_utente”, “email_venditore”, “nome_annuncio” uguali rispettivamente ai parametri presi in input, emailUtente, emailVenditore, nomeAnnuncio, e con un attributo “data_ora_annuncio “ uguale al seguente formato di concatenazioni stringhe java, dataOra.get(GregorianCalendar.YEAR)+"-"+ dataOra.get(GregorianCalendar.MONTH)+1+"- "+dataOra.get(GregorianCalendar.DAY_OF_MONTH)+" "+dataOra.get(GregorianCalendar.HOUR_OF_DAY)+":"+ dataOra.get(GregorianCalendar.MINUTE)+":"+ dataOra.get(GregorianCalendar.SECOND),

Metodo	public void deleteByKey(String emailUtente, String emailVenditore, String nomeAnnuncio, GregorianCalendar dataOra) throws SQLException
Descrizione	Il metodo setta l’attributo “eliminato” uguale a false del preferito dell’utente pertinente ai parametri in input.
Context	UtenteDAO:: deleteByKey (emailUtente :String, emailVenditore :String, nomeAnnuncio :String, dataOra :GregorianCalendar)
Precondizione	Sul database esiste un utente con attributo “email” uguale al parametro in input emailUtente.
Precondizione	Esiste sul database un annuncio ai preferiti con attributi “email_utente”, “email_venditore”, “nome_annuncio” uguali rispettivamente ai parametri presi in input, emailUtente, emailVenditore, nomeAnnuncio e con attributo “eliminato” uguale a false, e “data_ora_annuncio “ uguale al seguente formato di concatenazioni stringhe java, dataOra.get(GregorianCalendar.YEAR)+"-"+ dataOra.get(GregorianCalendar.MONTH)+1+"- "+dataOra.get(GregorianCalendar.DAY_OF_MONTH)+" "+dataOra.get(GregorianCalendar.HOUR_OF_DAY)+":"+ dataOra.get(GregorianCalendar.MINUTE)+":"+ dataOra.get(GregorianCalendar.SECOND).
Post condizione	Esiste sul database un annuncio ai preferiti con attributi “email_utente”, “email_venditore”, “nome_annuncio” uguali rispettivamente ai parametri presi in input, emailUtente, emailVenditore, nomeAnnuncio e con attributo “eliminato” uguale a true, e “data_ora_annuncio “ uguale al seguente formato di concatenazioni stringhe java, dataOra.get(GregorianCalendar.YEAR)+"-"+ dataOra.get(GregorianCalendar.MONTH)+1+"- "+dataOra.get(GregorianCalendar.DAY_OF_MONTH)+" "+dataOra.get(GregorianCalendar.HOUR_OF_DAY)+":"+ dataOra.get(GregorianCalendar.MINUTE)+":"+ dataOra.get(GregorianCalendar.SECOND).

Metodo	public synchronized void deleteAll(String email) throws SQLException
Descrizione	Il metodo setta l’attributo “eliminato” uguale a true, a tutti i preferiti sul database dell’utente con attributo “email” uguale al parametro in input email.
Context	UtenteDAO::deleteAll (email :String)
Precondizione	Sul database esiste un utente con attributo “email” uguale al parametro in input email.
Post condizione	Tutti gli annunci preferiti sul database con attributo “email_utente” uguale al parametro in input email, hanno l’attributo “eliminato” uguale a true.

OggettoDAO.java

La classe OggettoDAO si interfaccia sul database per eseguire le operazioni di inserimento, ricerca, di elimina e di ban di un annuncio.

Metodo	public synchronized OggettoBean doSave(OggettoBean c)
Descrizione	Il metodo serve per salvare le informazioni dell'annuncio contenute nell'oggetto preso in input, OggettoBean c, nel database.
Context	OggettoDAO::doSave(c :OggettoBean)
	c =! null.
	Sul database esiste un utente con attributo "email" uguale al parametro in input c.getEmail().
Precondizione	c.getCategoria() != null AND c.getDataOra() != null AND c. getDescrizione() != null AND c.getImmagine() != null AND c.getNome() != null AND c. getPrezzo() != null AND c. getRegione() != null.
Post condizione	Esiste un annuncio sul database con attributi "nome", "email_venditore", "immagine", "descrizione", "categoria", "regione", "prezzo" uguali rispettivamente ai parametri in input c.getNome(), c.getEmail(), c.getImmagine(), c.getDescrizione(), c.getCategoria(), c.getRegione(), c.getPrezzo().

Metodo	public synchronized ArrayList<OggettoBean> doSearch(String nome, String regione, String categoria)
Descrizione	Il metodo restituisce le informazioni riguardo agli annunci pertinenti ai parametri di ricerca presi in input se presenti nel database, in un oggetto ArrayList che è un insieme di oggetti di tipo OggettoBean, contenitore di annunci, altrimenti ritorna null.
Context	OggettoDAO::doSearch(nome :String, regione :String, categoria :String)
Precondizione	nome != null AND regione != null AND categoria != null.
Post condizione	ritorna in un insieme, gli annunci sul database,se presenti che hanno attributi "regione", "categoria" uguali rispettivamente ai parametri in input regione, categoria e l'attributo "nome" in cui è contenuto il parametro in input nome, e gli attributi "email_ban" = null e "eliminato" = false.

Metodo	public synchronized ArrayList<OggettoBean>doSearchNomeRegione(String nome, String regione)
Descrizione	Il metodo restituisce le informazioni riguardo agli annunci pertinenti ai parametri di ricerca presi in input se presenti nel database, in un oggetto ArrayList che è un insieme di oggetti di tipo OggettoBean, contenitore di annunci, altrimenti ritorna null.
Context	OggettoDao::doSearchNomeRegione(nome :String, regione :String)
Precondizione	nome != null AND regione != null.
Post condizione	Ritorna in un insieme, gli annunci sul database, se presenti, che hanno attributo "regione", uguale al parametro in input regione e l'attributo "nome" in cui è contenuto il parametro in input nome, e gli attributi "email_ban" = null e "eliminato" = false.

Metodo	public synchronized ArrayList<OggettoBean> doSearchNomeCategoria(String nome, String categoria)
Descrizione	Il metodo restituisce le informazioni riguardo agli annunci pertinenti ai parametri di ricerca presi in input se presenti nel database, in un oggetto ArrayList che è un insieme di oggetti di tipo OggettoBean, contenitore di annunci, altrimenti ritorna null.
Context	OggettoDAO::doSearchNomeCategoria (nome :String, categoria :String)
Precondizione	nome != null AND categoria != null.
Post condizione	ritorna in un insieme, gli annunci sul database, se presenti, che hanno attributo "categoria", uguale al parametro in input categoria e l'attributo "nome" in cui è contenuto il parametro in input nome, e gli attributi "email_ban" = null e "eliminato" = false.

Metodo	public synchronized ArrayList<OggettoBean> doSearchNome(String nome)
Descrizione	Il metodo restituisce le informazioni riguardo agli annunci pertinenti ai parametri di ricerca presi in input se presenti nel database, in un oggetto ArrayList che è un insieme di oggetti di tipo OggettoBean, contenitore di annunci, altrimenti null.
Context	OggettoDAO::doSearchNome (nome :String)
Precondizione	nome != null.
Post condizione	ritorna in un insieme, gli annunci sul database, se prensenti, che hanno l'attributo "nome" in cui è contenuto il parametro in input nome, e gli attributi "email_ban" = null e "eliminato" = false

Metodo	public synchronized OggettoBean doSearchByKey(String nome, String email_venditore, GregorianCalendar dataOra) throws SQLException
Descrizione	Il metodo restituisce in un OggettoBean l'annuncio del database pertinente ai parametri in input, se esiste, altrimenti null
Context	OggettoDAO::doSearchByKey (nome :String, email_venditore :String, dataOra :GregorianCalendar)

Precondizione	nome != null AND email_venditore != null AND dataOra != null.
Post condizione	Ritorna l'annuncio sul database che ha gli attributi "nome", "email_venditore" uguali ai parametri in input nome, email_venditore, e con l'attributo data_ora uguale al seguente formato di concatenazioni stringhe java, dataOra.get(GregorianCalendar.YEAR)+"-"+ dataOra.get(GregorianCalendar.MONTH)+1+"- "+dataOra.get(GregorianCalendar.DAY_OF_MONTH)+" "+dataOra.get(GregorianCalendar.HOUR_OF_DAY)+":"+ dataOra.get(GregorianCalendar.MINUTE)+":"+ dataOra.get(GregorianCalendar.SECOND), e con attributi "email_ban" = null e "eliminato" = false, se presente, altrimenti ritorna null.

Metodo	public synchronized ArrayList<OggettoBean> doSearchByEmail(String email_venditore)
Descrizione	Il metodo restituisce le informazioni riguardo agli annunci che ha pubblicato l'utente che ha l'email corrispondente al parametro in input email_venditore se presenti nel database, in un oggetto ArrayList che è un insieme di oggetti di tipo OggettoBean, contenitore di annunci, altrimenti null.
Context	OggettoDAO::doSearchByEmail(email_venditore :String)
Precondizione	email_venditore != null.
Post condizione	ritorna gli annunci dal database, se presenti, che hanno attributo "email_venditore" uguale al parametro in input email_venditore, e gli attributi "email_ban" = null e "eliminato" = false.

Metodo	public synchronized void deleteByKey(String email, String name, GregorianCalendar dataOra) throws SQLException
Descrizione	Il metodo elimina l'annuncio nel database, pertinente ai parametri in input.
Context	OggettoDAO::deleteByKey(email :String, name :String, dataOra :GregorianCalendar)
Precondizione	Esiste un annuncio sul database che ha gli attributi "nome", "email_venditore" uguali ai parametri in input name, email, e con l'attributo "data_ora" uguale al seguente formato di concatenazioni stringhe java, dataOra.get(GregorianCalendar.YEAR)+"-"+ dataOra.get(GregorianCalendar.MONTH)+1+"- "+dataOra.get(GregorianCalendar.DAY_OF_MONTH)+" "+dataOra.get(GregorianCalendar.HOUR_OF_DAY)+":"+ dataOra.get(GregorianCalendar.MINUTE)+":"+ dataOra.get(GregorianCalendar.SECOND), e con attributi "email_ban" = null e "eliminato" = false.
Post condizione	Esiste un annuncio sul database che ha gli attributi "nome", "email_venditore" uguali ai parametri in input name, email, e con l'attributo "data_ora" uguale al seguente formato di concatenazioni stringhe java, dataOra.get(GregorianCalendar.YEAR)+"-"+ dataOra.get(GregorianCalendar.MONTH)+1+"- "+dataOra.get(GregorianCalendar.DAY_OF_MONTH)+" "+dataOra.get(GregorianCalendar.HOUR_OF_DAY)+":"+ dataOra.get(GregorianCalendar.MINUTE)+":"+ dataOra.get(GregorianCalendar.SECOND), e con attributi "email_ban" = null e "eliminato" = true.

Metodo	public synchronized void banByKey(String email_gestore, String email_annuncio, String name, GregorianCalendar dataOra) throws SQLException
Descrizione	Il metodo setta l'attributo "ban" uguale al parametro in input email_gestore, sul database dell'annuncio pertinente ai parametri in input.
Context	OggettoDAO::banByKey (email_gestore :String, email_annuncio :String, name :String, dataOra :GregorianCalendar)
Precondizione	Esiste sul database un utente con attributo "email" uguale al parametro in input email_gestore.
	Esiste un annuncio sul database che ha gli attributi "nome", "email_venditore" uguali ai parametri in input name, email_annuncio, e con l'attributo "data_ora" uguale al seguente formato di concatenazioni stringhe java, dataOra.get(GregorianCalendar.YEAR)+"-"+ dataOra.get(GregorianCalendar.MONTH)+1+"- "+dataOra.get(GregorianCalendar.DAY_OF_MONTH)+" "+dataOra.get(GregorianCalendar.HOUR_OF_DAY)+":"+ dataOra.get(GregorianCalendar.MINUTE)+":"+ dataOra.get(GregorianCalendar.SECOND), e con attributi "email_ban" = null e "eliminato" = false.
Post condizione	Esiste un annuncio sul database che ha gli attributi "nome", "email_venditore" uguali ai parametri in input name, email, e con l'attributo "data_ora" uguale al seguente formato di concatenazioni stringhe java, dataOra.get(GregorianCalendar.YEAR)+"-"+ dataOra.get(GregorianCalendar.MONTH)+1+"- "+dataOra.get(GregorianCalendar.DAY_OF_MONTH)+" "+dataOra.get(GregorianCalendar.HOUR_OF_DAY)+":"+ dataOra.get(GregorianCalendar.MINUTE)+":"+ dataOra.get(GregorianCalendar.SECOND), e con attributi "email_ban" = email_gestore e "eliminato" = false.

Metodo	public synchronized void removeByKey(String email, String name, GregorianCalendar dataOra) throws SQLException{
Descrizione	Il metodo elimina dal database l'annuncio pertinente ai parametri in input.
Context	OggettoDAO::removeByKey(email :String, name :String, dataOra :GregorianCalendar)
Precondizione	Esiste un annuncio sul database che ha gli attributi "nome", "email_venditore" uguali ai parametri in input name, email, e con l'attributo "data_ora" uguale al seguente formato di concatenazioni stringhe java, dataOra.get(GregorianCalendar.YEAR)+"-"+ dataOra.get(GregorianCalendar.MONTH)+1+"- "+dataOra.get(GregorianCalendar.DAY_OF_MONTH)+" "+dataOra.get(GregorianCalendar.HOUR_OF_DAY)+":"+ dataOra.get(GregorianCalendar.MINUTE)+":"+ dataOra.get(GregorianCalendar.SECOND).
Post condizione	Non esiste un annuncio sul database che ha gli attributi "nome", "email_venditore" uguali ai parametri in input name, email, e con l'attributo "data_ora" uguale al seguente formato di concatenazioni stringhe java, dataOra.get(GregorianCalendar.YEAR)+"-"+ dataOra.get(GregorianCalendar.MONTH)+1+"- "+dataOra.get(GregorianCalendar.DAY_OF_MONTH)+" "+dataOra.get(GregorianCalendar.HOUR_OF_DAY)+":"+ dataOra.get(GregorianCalendar.MINUTE)+":"+ dataOra.get(GregorianCalendar.SECOND).

DriverManagerConnectionPool.java

La classe DriverManagerConnectionPool ha i metodi per connettersi al database.

Metodo	public static synchronized Connection getConnection() throws SQLException.
Descrizione	Il metodo stabilisce una connessione col database.

Metodo	public static synchronized void releaseConnection(Connection connection) throws SQLException
Descrizione	Il metodo rilascia una connessione.
Context	DriverManagerConnectionPool::releaseConnection(Connection connection)
Precondizione	Connection != null.
Precondizione	Connection è una connessione stabilità.
Post condizione	Connection è una connessione libera.

ChatDAO.java

La classe ChatDAO si interfaccia sul database per salvare e caricare messaggi e chat.

Metodo	public synchronized void doSave(MessaggioBean messaggio, String emailMittenteChat, String emailDestinatarioChat,String nomeAnnuncio, GregorianCalendar dataOraAnnuncio, int mittenteMessaggioNonLetto, int destinatarioMessaggioNonLetto) throws SQLException
Descrizione	Il metodo salva un messaggio pertinente ai parametri in input sul database, ed alla chat nel database relativa al massaggio in input al metodo, setta gli attributi relativi al messaggio non letto.
Context	ChatDAO::doSave (messaggio :MessaggioBean, emailMittenteChat :String, emailDestinatarioChat :String, nomeAnnuncio :String, dataOraAnnuncio :GregorianCalendar, mittenteMessaggioNonLetto :int, destinatarioMessaggioNonLetto :int)
Precondizione	Esiste sul database una chat con attributi “email_mittente”, “email_destinatario”, “nome_annuncio”, uguali ai parametri in input emailMittenteChat, emailDestinatarioChat, nomeAnnuncio, e con l’attributo “data_ora_annuncio” uguale al seguente formato di concatenazioni stringhe java, dataOraAnnuncio.get(GregorianCalendar.YEAR)+"-"+ dataOraAnnuncio.get(GregorianCalendar.MONTH)+"-"+ dataOraAnnuncio.get(GregorianCalendar.DAY_OF_MONTH)+" "+ dataOraAnnuncio.get(GregorianCalendar.HOUR_OF_DAY)+":"+ dataOraAnnuncio.get(GregorianCalendar.MINUTE)+":"+ dataOraAnnuncio.get(GregorianCalendar.SECOND).
Precondizione	messaggio.getDescrizione() != null AND messaggio.getEmailMittenteMessaggio() != null.
Precondizione	(messaggio.getEmailMittenteMessaggio().equals(emailMittenteChat) OR messaggio.getEmailMittenteMessaggio().equals(emailDestinatarioChat) .
Post condizione	Sul database è presente un messaggio con attributi “email_mittente_chat”, “email_destinatario_chat”, “nome_annuncio”, “descrizione”, “email_mittente_messaggio” uguali rispettivamente ai parametri in input emailMittenteChat, emailDestinatarioChat, nomeAnnuncio, messaggio.getDescrizione(), messaggio.getEmailMittenteMessaggio(), e con attributo “data_ora_annuncio” uguale al seguente formato di concatenazioni stringhe java, dataOraAnnuncio.get(GregorianCalendar.YEAR)+"-"+ dataOraAnnuncio.get(GregorianCalendar.MONTH)+"-"+ dataOraAnnuncio.get(GregorianCalendar.DAY_OF_MONTH)+" "+ dataOraAnnuncio.get(GregorianCalendar.HOUR_OF_DAY)+":"+ dataOraAnnuncio.get(GregorianCalendar.MINUTE)+":"+ dataOraAnnuncio.get(GregorianCalendar.SECOND).
Post condizione	Esiste sul database una chat con attributi “email_mittente”, “email_destinatario”, “nome_annuncio”, uguali ai parametri in input emailMittenteChat, emailDestinatarioChat, nomeAnnuncio, e con l’attributo “data_ora_annuncio” uguale al seguente formato di concatenazioni stringhe java, dataOraAnnuncio.get(GregorianCalendar.YEAR)+"-"+ dataOraAnnuncio.get(GregorianCalendar.MONTH)+"-"+ dataOraAnnuncio.get(GregorianCalendar.DAY_OF_MONTH)+" "+

	dataOraAnnuncio.get(GregorianCalendar.HOUR_OF_DAY)+":"+ dataOraAnnuncio.get(GregorianCalendar.MINUTE)+":"+ dataOraAnnuncio.get(GregorianCalendar.SECOND), e con attributi "mittente_messaggio_non_letto", "destinatario_messaggio_non_letto" uguali rispettivamente ai parametri in input mittenteMessaggioNonLetto e destinatarioMessaggioNonLetto

Metodo	public synchronized ArrayList<ChatBean> doSearchByEmail(String emailUtente) throws SQLException
Descrizione	Il metodo restituisce tutte le chat dell'utente sul database con attributo "email" uguale al parametro in input emailUtente.
Context	ChatDAO:: doSearchByEmail (emailUtente :String)
Precondizione	Sul database esiste un utente con attributo "email" uguale al parametro in input emailUtente.
Post condizione	Il metodo restituisce l'insieme di chat dal database, tale che il parametro in input emailUtente è uguale all'attributo della chat sul database "email_mittente" o uguale all'attributo "email_destinatario".

Metodo	public synchronized ArrayList<MessaggioBean> doSearchChatMessage(String emailMittenteChat, String emailDestinatarioChat, String nomeAnnuncio, GregorianCalendar dataOraAnnuncio, int settaMessaggioNonLetto, boolean mittenteMessaggioNonLetto) throws SQLException
Descrizione	Il metodo restituisce l'insieme di messaggi associati alla chat sul database pertinente ai parametri in input, e setta a codesta chat ,i messaggi non letti del mittente uguale al valore in input settaMessaggioNonLetto se il valore booleano in input mittenteMessaggioNonLetto è uguale a true, altrimenti lo setta ai messaggi non letti del destinatario.
Context	ChatDAO::doSearchChatMessage (emailMittenteChat :String, emailDestinatarioChat :String, nomeAnnuncio :String, dataOraAnnuncio :GregorianCalendar, settaMessaggioNonLetto :int, mittenteMessaggioNonLetto :boolean)
Precondizione	Esiste sul database una chat con attributi "email_mittente", "email_destinatario", "nome_annuncio", uguali ai parametri in input emailMittenteChat, emailDestinatarioChat, nomeAnnuncio, e con l'attributo "data_ora_annuncio" uguale al seguente formato di concatenazioni stringhe java, dataOraAnnuncio.get(GregorianCalendar.YEAR)+"-"+ dataOraAnnuncio.get(GregorianCalendar.MONTH)+1+"-"+ dataOraAnnuncio.get(GregorianCalendar.DAY_OF_MONTH)+" "+ dataOraAnnuncio.get(GregorianCalendar.HOUR_OF_DAY)+":"+ dataOraAnnuncio.get(GregorianCalendar.MINUTE)+":"+ dataOraAnnuncio.get(GregorianCalendar.SECOND).
Post condizione	Restituisce l'insieme di messaggi dal database con attributi "email_mittente_chat", "email_destinatario_chat", "nome_annuncio", uguali rispettivamente ai parametri in input emailMittenteChat, emailDestinatarioChat, nomeAnnuncio, e con attributo "data_ora_annuncio" uguale al seguente formato di concatenazioni stringhe java, dataOraAnnuncio.get(GregorianCalendar.YEAR)+"-"+ dataOraAnnuncio.get(GregorianCalendar.MONTH)+1+"-"+ dataOraAnnuncio.get(GregorianCalendar.DAY_OF_MONTH)+" "+ dataOraAnnuncio.get(GregorianCalendar.HOUR_OF_DAY)+":"+

	dataOraAnnuncio.get(GregorianCalendar.MINUTE)+":"+ dataOraAnnuncio.get(GregorianCalendar.SECOND).
Post condizione	La chat sul database che ha gli attributi “email_mittente”, “email_destinatario”, “nome_annuncio” uguali rispettivamente ai parametri in input emailMittenteChat, emailDestinatarioChat, nomeAnnuncio, e con attributo “data_ora_annuncio” uguale al seguente formato di concatenazioni stringhe java, dataOraAnnuncio.get(GregorianCalendar.YEAR)+"-"+ dataOraAnnuncio.get(GregorianCalendar.MONTH)+1+"-"+ dataOraAnnuncio.get(GregorianCalendar.DAY_OF_MONTH)+" "+ dataOraAnnuncio.get(GregorianCalendar.HOUR_OF_DAY)+":"+ dataOraAnnuncio.get(GregorianCalendar.MINUTE)+":"+ dataOraAnnuncio.get(GregorianCalendar.SECOND), ha l’attributo “mittente_messaggio_non_letto” uguale al valore settaMessaggioNonLetto se il valore in input mittenteMessaggioNonLetto è uguale a true, altrimenti se è uguale a false, allora ha l’attributo “destinatario_messaggio_non_letto” uguale al valore settaMessaggioNonLetto.

Metodo	public synchronized void iniziaNuovaChat(String emailMittenteChat, String emailDestinatarioChat, String nomeAnnuncio, GregorianCalendar dataOraAnnuncio) throws SQLException
Descrizione	Il metodo inizia una nuova chat tra gli utenti che hanno attributo “email” sul database uguale al parametro in input emailMittenteChat e emailDestinatarioChat.
Context	ChatDAO:: iniziaNuovaChat(emailMittenteChat :String, emailDestinatarioChat :String, nomeAnnuncio :String, dataOraAnnuncio :GregorianCalendar)
Precondizione	emailMittenteChat != null AND emailDestinatarioChat != null AND nomeAnnuncio != null AND dataOraAnnuncio != null.
Precondizione	Non esiste sul database una chat con attributi “email_mittente”, “email_destinatario”, “nome_annuncio”, uguali ai parametri in input emailMittenteChat, emailDestinatarioChat, nomeAnnuncio, e con l’attributo “data_ora_annuncio” uguale al seguente formato di concatenazioni stringhe java, dataOraAnnuncio.get(GregorianCalendar.YEAR)+"-"+ dataOraAnnuncio.get(GregorianCalendar.MONTH)+"-"+ dataOraAnnuncio.get(GregorianCalendar.DAY_OF_MONTH)+" "+ dataOraAnnuncio.get(GregorianCalendar.HOUR_OF_DAY)+":"+ dataOraAnnuncio.get(GregorianCalendar.MINUTE)+":"+ dataOraAnnuncio.get(GregorianCalendar.SECOND).
Precondizione	Sul database esiste un utente con attributo “email” uguale al parametro in input emailMittenteChat.
Precondizione	Sul database esiste un utente con attributo “email” uguale al parametro in input emailDestinatarioChat.
Precondizione	Esiste un annuncio sul database che ha gli attributi “nome”, “email_venditore” uguali ai parametri in input nomeAnnuncio, emailDestinatarioChat, e con l’attributo “data_ora” uguale al seguente formato di concatenazioni stringhe java, dataOraAnnuncio.get(GregorianCalendar.YEAR)+"-"+ dataOraAnnuncio.get(GregorianCalendar.MONTH)1+"-"+ dataOraAnnuncio.get(GregorianCalendar.DAY_OF_MONTH)+" "+ dataOraAnnuncio.get(GregorianCalendar.HOUR_OF_DAY)+":"+ dataOraAnnuncio.get(GregorianCalendar.MINUTE)+":"+ dataOraAnnuncio.get(GregorianCalendar.SECOND), e con attributi “email_ban” = null e “eliminato” = true.
Post condizione	Esiste sul database una chat con attributi “email_mittente”, “email_destinatario”, “nome_annuncio”, uguali ai parametri in input emailMittenteChat, emailDestinatarioChat, nomeAnnuncio, e con

	l'attributo "data_ora_annuncio" uguale al seguente formato di concatenazioni stringhe java, dataOraAnnuncio.get(GregorianCalendar.YEAR)+"-"+ dataOraAnnuncio.get(GregorianCalendar.MONTH)+"-"+ dataOraAnnuncio.get(GregorianCalendar.DAY_OF_MONTH)+" "+ dataOraAnnuncio.get(GregorianCalendar.HOUR_OF_DAY)+":"+ dataOraAnnuncio.get(GregorianCalendar.MINUTE)+":"+ dataOraAnnuncio.get(GregorianCalendar.SECOND).

3.7 Package model.beans

UtenteBean.java

La classe è utilizzata per contenere le informazioni dell'utente e contiene i vari metodi per prelevarle e modificarle.

Metodo	public int getAmministratore()
Descrizione	Il metodo restituisce un valore 1 oppure 0 che indica rispettivamente che l'utente è un amministratore o meno.

Metodo	public int getAnno_nascita()
Descrizione	Il metodo restituisce l'anno di nascita dell'utente.

Metodo	public String getEmail()
Descrizione	Il metodo restituisce l'email dell'utente.

Metodo	public String getNome()
Descrizione	Il metodo restituisce il nome dell'utente.

Metodo	public String getPass()
Descrizione	Il metodo restituisce la password dell'utente.

Metodo	public String getRegione()
Descrizione	Il metodo restituisce la regione dell'utente.

Metodo	public String getSesso()
Descrizione	Il metodo restituisce il sesso dell'utente.

Metodo	public long getTelefono()
Descrizione	Il metodo restituisce il numero di telefono del'utente.

Metodo	public void setAmministratore(int amministratore)
Descrizione	Il metodo setta o revoca l'incarico di amministratore all'utente.
Context	UtenteBean::setAmministratore(amministratore :int)
Post condizione	self.getAmministratore() = amministratore.

Metodo	public void setAnno_nascita(String anno)
Descrizione	Il metodo setta l'anno di nascita all'utente.
Context	UtenteBean::setAnno_nascita(anno :String)
Precondizione	La stringa anno deve essere un numero intero.
Precondizione	anno != null.
Post condizione	self.getAnno_nascita() = Integer.parseInt(anno).

Metodo	public void setEmail(String email)

Descrizione	Il metodo modifica l'email dell'utente.
Context	UtenteBean::setEmail(email :String)
Precondizione	email != null.
Post condizione	self.getEmail().equals(email).

Metodo	public void setNome(String nome)
Descrizione	Il metodo modifica il nome dell'utente.
Context	UtenteBean::setNome(nome :String)
Precondizione	nome != null.
Post condizione	self.getNome().equals(nome).

Metodo	public void setPass(String pass)
Descrizione	Il metodo modifica la password dell'utente.
Context	UtenteBean::setPass(pass :String)
Precondizione	pass != null.
Post condizione	self.getPass().equals(pass).

Metodo	public void setRegione(String regione)
Descrizione	Il metodo modifica la regione dell'utente.
Context	UtenteBean::setRegione(regione :String)
Precondizione	regione != null.
Post condizione	self.getRegione().equals(regione).

Metodo	public void setSesso(String sesso)
Descrizione	Il metodo modifica il sesso dell'utente.

Context	UtenteBean::setSesso(sesso :String)
Precondizione	sesso != null.
Post condizione	self.getSesso.equals(sesso).

Metodo	public void setTelefono(long telefono)
Descrizione	Il metodo modifica il numero di telefono.
Context	UtenteBean::setTelefono(telefono :long)
Post condizione	self.getTelefono() = telefono.

Metodo	public String getEmail_ban()
Descrizione	Ritorna l'email che ha bannato l'account, se nussuno ha bannato l'account ritorna null.

Metodo	public void setEmail_ban(String email_ban)
Descrizione	Setta l'istanza l'email_ban dell'account al parametro preso in input.
Context	UtenteBean::setEmail_ban(email_ban :String)
Precondizione	email_ban != null.
Post condizione	self.getEmail_ban() = email_ban.

Metodo	public boolean getEliminato()
Descrizione	Restituisce il valore della varibiale d'istanza eliminato.

Metodo	public void setEliminato(boolean eliminato)
Descrizione	Setta l'istanza eliminato dell'account al parametro preso in input.
Context	UtenteBean:: setEliminato(eliminato :boolean)
Post condizione	self.getEliminato = eliminato.

OggettoBean.java

La classe è utilizzata per contenere le informazioni degli annunci e contiene i vari metodi per prelevarle e modificarle.

Metodo	public String getCategoria()
Descrizione	Il metodo restituisce la categoria in cui è stato pubblicato l'annuncio.

Metodo	public String getDescrizione()
Descrizione	Il metodo restituisce la descrizione dell'annuncio.

Metodo	public String getEmail()
Descrizione	Il metodo restituisce l'email dell'utente che ha pubblicato l'annuncio.

Metodo	public String getImmagine()
Descrizione	Il metodo restituisce l'url dell'immagine dell'annuncio.

Metodo	public String getNome()
Descrizione	Il metodo restituisce il nome dell'annuncio.

Metodo	public String getNome_proprietario()
Descrizione	Il metodo restituisce il nome di chi ha pubblicato l'annuncio.

Metodo	public String getNumero_proprietario()

Descrizione	Il metodo restituisce il nome di chi ha pubblicato l'annuncio.

Metodo	public String getNumero_proprietario()
Descrizione	Il metodo restituisce il numero di telefono di chi ha pubblicato l'annuncio.

Metodo	public int getPrezzo()
Descrizione	Il metodo restituisce il prezzo dell'annuncio.

Metodo	public String getRegione()
Descrizione	Il metodo restituisce la regione di chi ha pubblicato l'annuncio.

Metodo	public GregorianCalendar getDataOra ()
Descrizione	Il metodo restituisce l'ora in cui è stato pubblicato l'annuncio.

Metodo	public boolean getEliminato()
Descrizione	Il metodo restituisce il valore della variabile d'istanza eliminato.

Metodo	public void setCategoria(String categoria)
Descrizione	Il metodo modifica la categoria dell'annuncio.
Context	OggettoBean::setCategoria(categoria :String)
Precondizione	categoria != null.
Post condizione	self.getCategoria() = categoria.

Metodo	public void setDescrizione(String descrizione)
Descrizione	Il metodo modifica la descrizione dell'annuncio.

Context	OggettoBean::setDescrizione(descrizione :String)
Precondizione	descrizione != null.
Post condizione	self.getDescrizione() = descrizione.

Metodo	public void setEmail(String email)
Descrizione	Il metodo modifica l'email dell'annuncio.
Context	OggettoBean::setEmail(email :String)
Precondizione	email!= null.
Post condizione	self. getEmail() = email.

Metodo	public void setImmagine(String immagine)
Descrizione	Il metodo modifica l'url dell'annuncio.
Context	OggettoBean::setImmagine(immagine :String)
Precondizione	immagine != null.
Post condizione	self.getImmagine() = immagine.

Metodo	public void setNome(String nome)
Descrizione	Il metodo modifica il nome dell'annuncio.
Context	OggettoBean:: setNome (nome :String)
Precondizione	nome != null.
Post condizione	self.getNome() = nome.

Metodo	public void setNome_proprietario(String nome_proprietario)
Descrizione	Il metodo modifica il nome di chi ha pubblicato l'annuncio.
Context	OggettoBean:: setNome_proprietario(nome_proprietario :String)

Precondizione	nome_proprietario!= null.
Post condizione	self.getNome_proprietario() = nome_proprietario.

Metodo	public void setNumero_proprietario(String numero_proprietario)
Descrizione	Il metodo modifica il numero di telefono di chi ha pubblicato l'annuncio.
Context	OggettoBean:: setNumero_proprietario(numero_proprietario :String)
Precondizione	numero_proprietario!= null.
Post condizione	self.getNumero_proprietario() = numero_proprietario.

Metodo	public void setPrezzo(int prezzo)
Descrizione	Il metodo modifica il prezzo dell'annuncio.
Context	OggettoBean:: setPrezzo(prezzo :int)
Post condizione	self.getPrezzo() = prezzo.

Metodo	public void setRegione(String regione)
Descrizione	Il metodo modifica la regione dell'annuncio.
Context	OggettoBean:: setRegione(regione :String)
Precondizione	regione!= null.
Post condizione	self.getRegione() = regione.

Metodo	public void setDataOra(GregorianCalendar dataOra)
Descrizione	Il metoso setta la data e l'ora di pubblicazione dell'annuncio a quella presa in input.
Context	OggettoBean::setDataOra(dataOra :GregorianCalendar)
Precondizione	dataOra != null.
Post condizione	self.getDataOra() = dataOra.

Metodo	public void setEliminato(boolean eliminato)
Descrizione	Il metodo setta al valora preso in input la variabile d'istanze eliminato dell'oggetto.
Context	OggettoBean:: setEliminato(eliminato :boolean)
Post condizione	self.getEliminato = eliminato.

MessaggioBean.java

La classe offre i metodi per settare e prelevare i vari parametri di un messaggio.

Metodo	public GregorianCalendar getDataOraInvioMessaggio()
Descrizione	Il metodo restituisce un oggetto GregorianCalendar che indica quando è stato inviato il messaggio.

Metodo	public String getDescrizione()
Descrizione	Il metodo restituisce la descrizione del messaggio.

Metodo	public String getEmailDestinatarioChat()
Descrizione	Il metodo restituisce l'email del destinatario della chat.

Metodo	public String getEmailMittenteChat()
Descrizione	Il metodo restituisce l'email del mittente della chat.

Metodo	public String getEmailMittenteMessaggio()
Descrizione	Il metodo restituisce l'email del mittente del messaggio.

Metodo	public String getNomeAnnuncioChat()
Descrizione	Il metodo restituisce il nome dell'annuncio sulla quale è iniziata la chat.

Metodo	public void setDataOraInvioMessaggio(GregorianCalendar dataOraInvioMessaggio)
Descrizione	Il metodo setta la data e l'ora d'invio del messaggio al parametro in input.

Context	MessaggioBean::setDataOraInvioMessaggio (dataOraInvioMessaggio :GregorianCalendar).
Precondizione	dataOraInvioMessaggio != null.
Post condizione	self.getDataOraInvioMessaggio() = dataOraInvioMessaggio.

Metodo	public void setDescrizione(String descrizione)
Descrizione	Il metodo setta la descrizione del messaggio al parametro in input.
Context	MessaggioBean::setDescrizione (descrizione :String).
Precondizione	descrizione != null.
Post condizione	self.getDescrizione() = descrizione.

Metodo	public void setEmailDestinatarioChat(String emailDestinatarioChat)
Descrizione	Il metodo setta l'email del destinatario della chat al parametro in input.
Context	MessaggioBean::setEmailDestinatarioChat (emailDestinatarioChat :String).
Precondizione	emailDestinatarioChat != null.
Post condizione	self.getEmailDestinatarioChat() = emailDestinatarioChat.

Metodo	public void setEmailMittenteChat(String emailMittenteChat)
Descrizione	Il metodo setta l'email mittente chat al parametro in input.
Context	MessaggioBean::setEmailMittenteChat (emailMittenteChat :String).
Precondizione	emailMittenteChat != null.
Post condizione	self.getEmailMittenteChat() = emailMittenteChat.

Metodo	public void setEmailMittenteMessaggio(String emailMittenteMessaggio)
Descrizione	Il metodo setta l'email mittente del messaggio al parametro in input.
Context	MessaggioBean::setEmailMittenteMessaggio (emailMittenteMessaggio :String)

Precondizione	emailMittenteMessaggio != null.
Post condizione	self.getEmailMittenteMessaggio() = emailMittenteMessaggio.

Metodo	public void setNomeAnnuncioChat(String nomeAnnuncioChat)
Descrizione	Il metodo setta il nome annuncio della chat al parametro in input nomeAnnuncioChat.
Context	MessaggioBean::setNomeAnnuncioChat (nomeAnnuncioChat :String)
Precondizione	nomeAnnuncioChat != null.
Post condizione	self.getNomeAnnuncioChat() = nomeAnnuncioChat.

ChatBean.java

La classe offre i metodi per settare e prelevare le informazioni della chat.

Metodo	public GregorianCalendar getDataOraAnnuncio()
Descrizione	Il metodo restituisce la data e l'ora dell'annuncio.

Metodo	public int getDestinatarioMessaggioNonLetto()
Descrizione	Il metodo restituisce il numero di messaggi non letti.

Metodo	public String getEmailDestinatario()
Descrizione	Il metodo restituisce l'email del destinatario.

Metodo	public String getEmailMittente()
Descrizione	Il metodo restituisce l'email mittente.

Metodo	public String getImmagine ()
Descrizione	Il metodo restituisce l'url dell'immagine.

Metodo	public int getMittenteMessaggioNonLetto ()
Descrizione	Il metodo restituisce il numero di messaggi non letti dal mittente.

Metodo	public String getNomeAnnuncio()
Descrizione	Il metodo restituisce il nome dell'annuncio.

Metodo	public String getNomeMittenteChat ()
Descrizione	Il metodo restituisce il nome del mittente della chat.

Metodo	public String getNomeVenditoreAnnuncio ()
Descrizione	Il metodo restituisce il nome del venditore dell'annuncio, cioè il destinatario chat.

Metodo	public void setDataOraAnnuncio (GregorianCalendar dataOraAnnuncio)
Descrizione	Il metodo setta la data e l'ora dell'annuncio chat, al parametro in input.
Context	ChatBean::setDataOraAnnuncio (dataOraAnnuncio :GregorianCalendar)
Precondizione	dataOraAnnuncio != null.
Post condizione	self.getDataOraAnnuncio() = dataOraAnnuncio.

Metodo	public void setDestinatarioMessaggioNonLetto (int destinatarioMessaggioNonLetto)
Descrizione	Il metodo setta il numero dei messaggi non letti al destinatario, uguale al parametro in input.
Context	Context ChatBean::setDestinatarioMessaggioNonLetto (destinatarioMessaggioNonLetto :int)
Post condizione	Post: self.getDestinatarioMessaggioNonLetto ()= destinatarioMessaggioNonLetto.

Metodo	public void setEmailDestinatario (String emailDestinatario)
Descrizione	Il metodo setta l'email del destinatario al parametro in input.
Context	ChatBean::setEmailDestinatario (emailDestinatario :String)
Precondizione	emailDestinatario != null.
Post condizione	self.getEmailDestinatario() = emailDestinatario.

Metodo	public void setEmailMittente (String emailMittente)
Descrizione	Il metodo setta l'email mittente al parametro in input.
Context	ChatBean::setEmailMittente (emailMittente :String)
Precondizione	emailMittente != null.
Post condizione	self.getEmailMittente() = emailMittente.

Metodo	public void setImmagine (String immagine)
Descrizione	Il metodo setta l'url dell'immagine al parametro in input.
Context	ChatBean::setImmagine (immagine :String)
Precondizione	immagine != null.
Post condizione	self.getImmagine() = immagine.

Metodo	public void setMittenteMessaggioNonLetto (int mittenteMessaggioNonLetto)
Descrizione	Il metodo setta il numero di messaggi non letti del mittente uguale al parametro in input.
Context	ChatBean:: setMittenteMessaggioNonLetto (mittenteMessaggioNonLetto :int)
Post condizione	self.getMittenteMessaggioNonletto() = mittenteMessaggioNonLetto.

Metodo	public void setNomeAnnuncio (String nomeAnnuncio)
Descrizione	Il metodo setta il nome dell'annuncio della chat uguale al parametro in input.
Context	ChatBean::setNomeAnnuncio (nomeAnnuncio :String)
Precondizione	nomeAnnuncio != null.
Post condizione	self.getNomeAnnuncio() = nomeAnnuncio

Metodo	public void setNomeMittenteChat (String nomeMittenteChat)
Descrizione	Il metodo setta il nome mittente della chat al parametro in input.

Context	ChatBean::setNomeMittenteChat (nomeMittenteChat :String)
Precondizione	nomeMittenteChat != null.
Post condizione	self.getNomeMittenteChat() = nomeMittenteChat.

Metodo	public void setNomeVenditoreAnnuncio (String nomeVenditoreAnnuncio)
Descrizione	Il metodo setta il nome del venditore dell'annuncio, cioè il destinatario chat, uguale al parametro in input.
Context	ChatBean::setNomeVenditoreAnnuncio (nomeVenditoreAnnuncio :String)
Precondizione	nomeVenditoreAnnuncio != null.
Post condizione	self.getNomeVenditoreAnnuncio() = nomeVenditoreAnnuncio.

4. Class diagram