UNIVERSITÀ DEGLI STUDI DI SALERNO
ANNO ACCADEMICO 2016/2017

CROWDMINE

Object Design Document

Versione 1.0

TOP MANAGER:
Prof. Andrea De Lucia

PROJECT MANAGER:
Antonio Luca D'Avanzo
Fabiano Pecorelli

Top Manager:

Nome
Prof. De Lucia Andrea

Project Manager:

Nome	Matricola
Antonio Luca D’Avanzo	051210 2502
Fabiano Pecorelli	052250 0421

Partecipanti:

Nome	Matricola
Severino Ammirati	051210 2898
Andrea Buonaguro	051210 2490
Angelo Caputo	051210 2204
Ferdinando D’Avino	051210 2360
Paolo Di Filippo	051210 3120
Alfredo Fiorillo	051210 1930
Dario Galiani	051210 2276
Giovanni Leo	051210 3062
Fabricio Nicolas Madaio	051210 2840
Vincenzo Noviello	051210 3198
Andrea Sarto	051210 2912

Lino Sarto	051210 2348
Giorgio Vitiello	051210 2318

Revision History:

Data	Versione	Descrizione	Autore
10/12/2016	1.0	Stesura del Object Design Document	Membri del Team

Indice

1. Introduzione

1.1 Object Design Trade-offs

Dopo la realizzazione dei documenti RAD e SDD abbiamo descritto in linea di massima quello che sarà il nostro sistema e quindi i nostri obiettivi, tralasciando gli aspetti implementativi. Il seguente documento ha lo scopo di produrre un modello capace di integrare in modo coerente e preciso tutte le funzionalità individuate nelle fasi precedenti. In particolare definisce le interfacce delle classi, le operazioni, i tipi, gli argomenti e la signature dei sottosistemi definiti nel System Design.
Inoltre sono specificati i trade-off e le linee guida.

Comprensibilità vs Tempo:

Il codice deve essere quanto più comprensibile possibile per facilitare la fase di testing ed eventuali future modifiche. Il codice sarà quindi accompagnato da commenti che ne semplifichino la comprensione. Ovviamente questa caratteristica aggiungerà un incremento di tempo allo sviluppo del nostro progetto.

Prestazioni vs Costi:

Essendo il progetto sprovvisto di budget, al fine di mantenere prestazioni elevate, per alcune funzionalità verranno utilizzati dei template open source esterni in particolare Bootstrap.

Interfaccia vs Usabilità:

L'interfaccia grafica è stata realizzata in modo da essere molto semplice, chiara e concisa, fa uso di form e pulsanti disposti in maniera da rendere semplice l'utilizzo del sistema da parte dell'utente finale.

Sicurezza vs Efficienza:

La sicurezza, come descritto nei requisiti non funzionali del RAD, rappresenta uno degli aspetti importanti del sistema. Tuttavia, dati i tempi di sviluppo molto limitati, ci limiteremo ad implementare sistemi di sicurezza basati su username e password degli utenti.

1.2 Linee Guida per la Documentazione delle Interfacce

Gli sviluppatori dovranno seguire alcune linee guida per la scrittura del codice:

Naming Convention

- E' buona norma utilizzare nomi:

    1. Descrittivi
    2. Pronunciabili
    3. Di uso comune
    4. Lunghezza medio-corta
    5. Non abbreviati
    6. Evitando la notazione ungherese
    7. Utilizzando solo caratteri consentiti (a-z, A-Z, 0-9)

Variabili:

- I nomi delle variabili devono cominciare con una lettera minuscola, e le parole seguenti con la lettera maiuscola. Quest'ultime devono essere dichiarate ad inizio blocco, solamente una per riga e devono essere tutte allineate per facilitare la leggibilità.

Esempio: elaboratoSessioneId

- E' inoltre possibile, in alcuni casi, utilizzare il carattere **underscore** "_", ad esempio quando utilizziamo delle variabili costanti oppure quando vengono utilizzate delle proprietà statiche.

Esempio: CREATE_ARGOMENTO

Metodi:

- I nomi dei metodi devono cominciare con una lettera minuscola, e le parole seguenti con la lettera maiuscola. Il nome del metodo tipicamente consiste di un verbo che identifica una azione, seguito dal nome di un oggetto. I nomi dei metodi per l'accesso e la modifica delle variabili dovranno essere del tipo getNomeVariabile() e setNomeVariabile(). Le variabili dei metodi devono essere dichiarate appena prima del loro utilizzo e devono servire per un solo scopo, per facilitare la leggibilità. Esistono però casi particolari come ad esempio nell'implementazione dei model, dove viene utilizzata l'interfaccia CRUD.

Esempio: getId(), setId()

- I commenti dei metodi devono essere raggruppati in base alla loro funzionalità, la descrizione dei metodi deve apparire prima di ogni dichiarazione di metodo, e deve descriverne lo scopo. Deve includere anche informazioni sugli argomenti, sul valore di ritorno, e se applicabile, sulle eccezioni.

Classi e pagine :

- I nomi delle classi e delle pagine devono cominciare con una lettera maiuscola, e anche le parole seguenti all'interno del nome devono cominciare con una lettera maiuscola. I nomi di quest'ultime devono fornire informazioni sul loro scopo.

Esempio: RispostaMultipla.php

Ogni file sorgente php contiene una singola classe e dev'essere strutturato in un determinato modo:

- Una breve introduzione alla classe

  L'introduzione indica: l'autore, la versione e la data.

```
/**
* sommario dello scopo della classe.
*
* @author nome dell'autore
* @version numero di versione della classe
* @since data d'implementazione
*/
```

- L'istruzione include che permette di importare all'interno della classe gli altri oggetti che la classe utilizza.
- La dichiarazione di classe caratterizzata da:
  1. Dichiarazione della classe pubblica
  2. Dichiarazioni di costanti
  3. Dichiarazioni di variabili di classe
  4. Dichiarazioni di variabili d'istanza
  5. Costruttore
  6. Commento e dichiarazione metodi.

ESEMPIO:

```php
<?php

/**
* Created by PhpStorm.
* User: Angelo
* Date: 30/11/2016
* Time: 20:40
*/

class tipoNotifica{
   const INSERIMENTO = "inserimento";
   const RISOLUZIONE = "risoluzione";
   const DECISIONE = "decisione";
}
```

```php
class Notifica implements JsonSerializable{
    private $id;
    private $data;
    private $tipo;
    private $info;
    private $letto;

    /**
     * Notifica constructor.
     * @param $id
     * @param $data
     * @param $tipo
     * @param $info
     * @param $letto
     */
    public function __construct($data, $tipo, $info, $letto, $id =null){
        $this->id = $id;
        $this->data = $data;
        $this->tipo = $tipo;
        $this->info = $info;
        $this->letto = $letto;
    }

    /**
     * @return mixed
     */
    public function getId(){
        return $this->id;
    }

    /**
     * @return mixed
     */
    public function getData(){
        return $this->data;
    }

    /**
     * @return mixed
     */
    public function getTipo(){
        return $this->tipo;
    }

    /**
     * @return mixed
     */
    public function getInfo(){
        return $this->info;
    }
```

```
    /**
     * @return mixed
     */
    public function getLetto(){
        return $this->letto;
    }

    /**
     * @param mixed $data
     */
    public function setData($data){
        $this->data = $data;
    }

    /**
     * @param mixed $tipo
     */
    public function setTipo($tipo){
        $this->tipo = $tipo;
    }

    /**
     * @param mixed $info
     */
    public function setInfo($info){
        $this->info = $info;
    }

    /**
     * @param mixed $letto
     */
    public function setLetto($letto){
        $this->letto = $letto;
    }


    function jsonSerialize(){
        return "{id: $this->getId(), data: $this->getData(), tipo:
$this->getTipo(), letto: $this->getLetto(), info: $this->getLetto()}";
    }

}
```

1.3 Definizioni, acronimi e abbreviazioni

Acronimi:

- RAD: Requirements Analysis Document
- SDD: System Design Document
- ODD: Object Design Document
- CRUD: Create Read Update Delete

Abbreviazioni:

- DB: DataBase

1.4 Riferimenti

- B. Bruegge, A. H. Dutoit, Object Oriented Software Engineering - Using UML, Pattern and Java, Prentice Hall, 3rd edition, 2009
- Documento SDD del progetto CrowdMine
- Documento RAD del progetto CrowdMine

2. Packages

La gestione del nostro sistema è suddivisa in tre livelli (three-tier):

- Interface layer
- Application Logic layer
- Storage layer

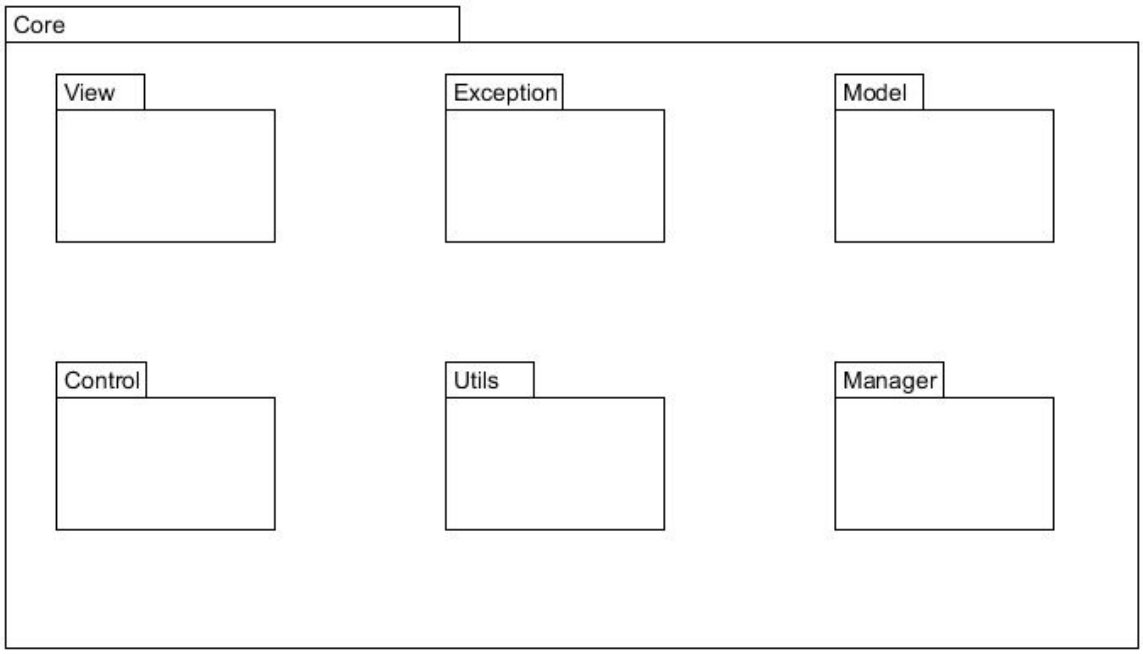
Il package CrowdMine contiene sottopackage che a loro volta inglobano classi atte alla gestione delle richieste utente. Le classi contenute nel package svolgono il ruolo di gestore logico del sistema.

Interface layer	Rappresenta l'interfaccia del sistema, ed offre la possibilità all'utente di interagire con quest'ultimo, offrendo sia la possibilità di inviare, in input, che di visualizzare, in output, dati.

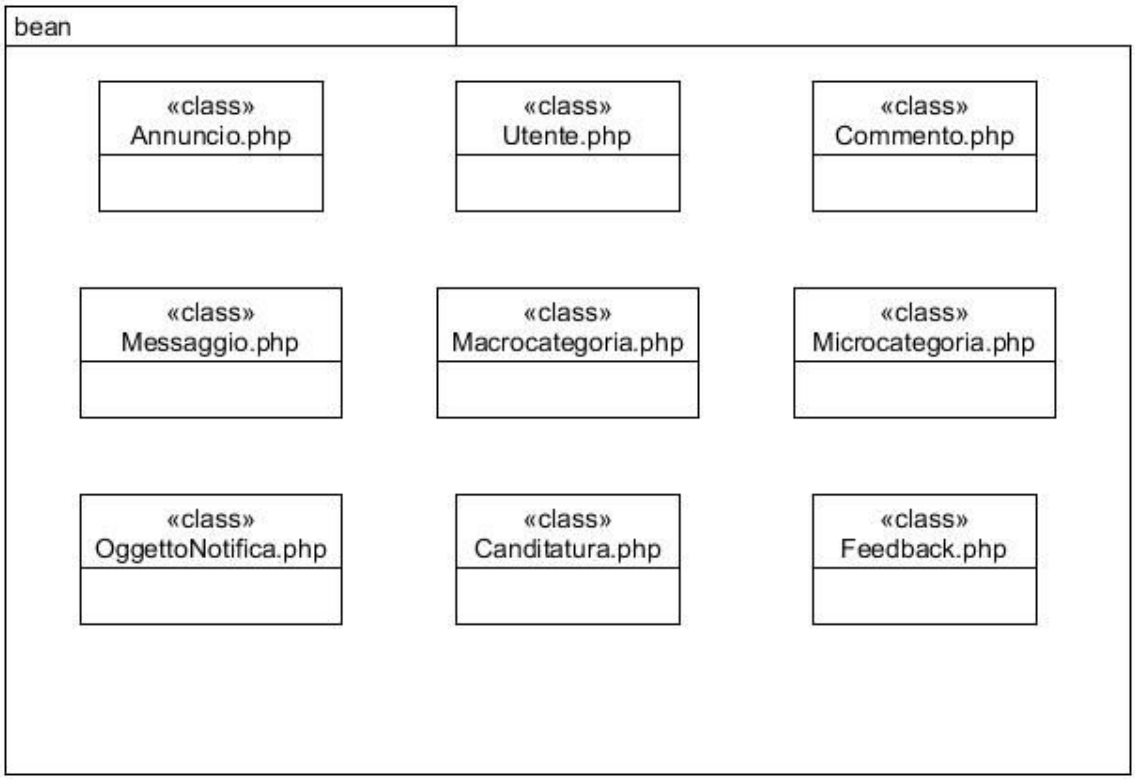
Application Logic layer	Ha il compito di elaborare i dati da inviare al client, e spesso grazie a delle richieste fatte al database, tramite lo Storage Layer, accede ai dati persistenti. Si occupa di varie gestioni quali: **1. Gestione Utente** **2. Gestione Annunci** **3. Gestione Notifiche e messaggi** **4. Gestione Feedback** **5. Gestione Categorie**

Storage layer	Ha il compito di memorizzare i dati sensibili del sistema, utilizzando un DBMS, inoltre riceve le varie richieste dall' Application Logic layer inoltrandole al DBMS e restituendo i dati richiesti.

2.1 Packages core

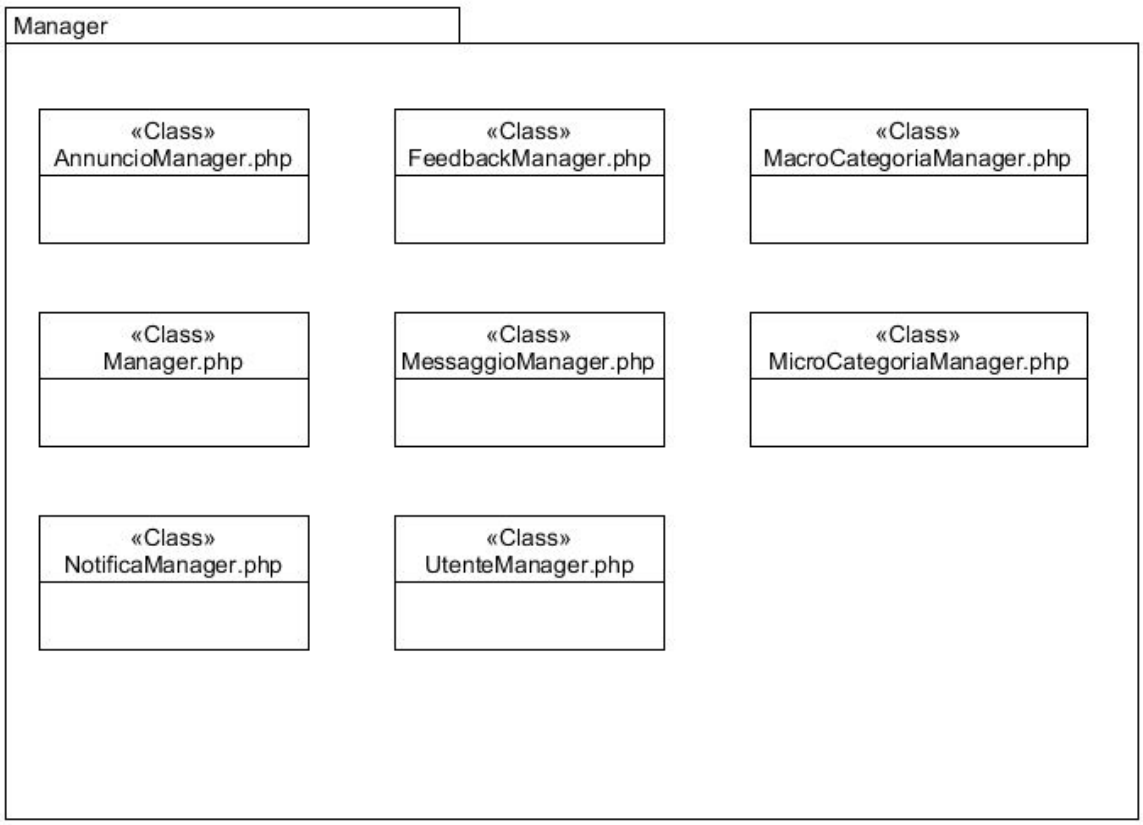


2.1.1 Packages model



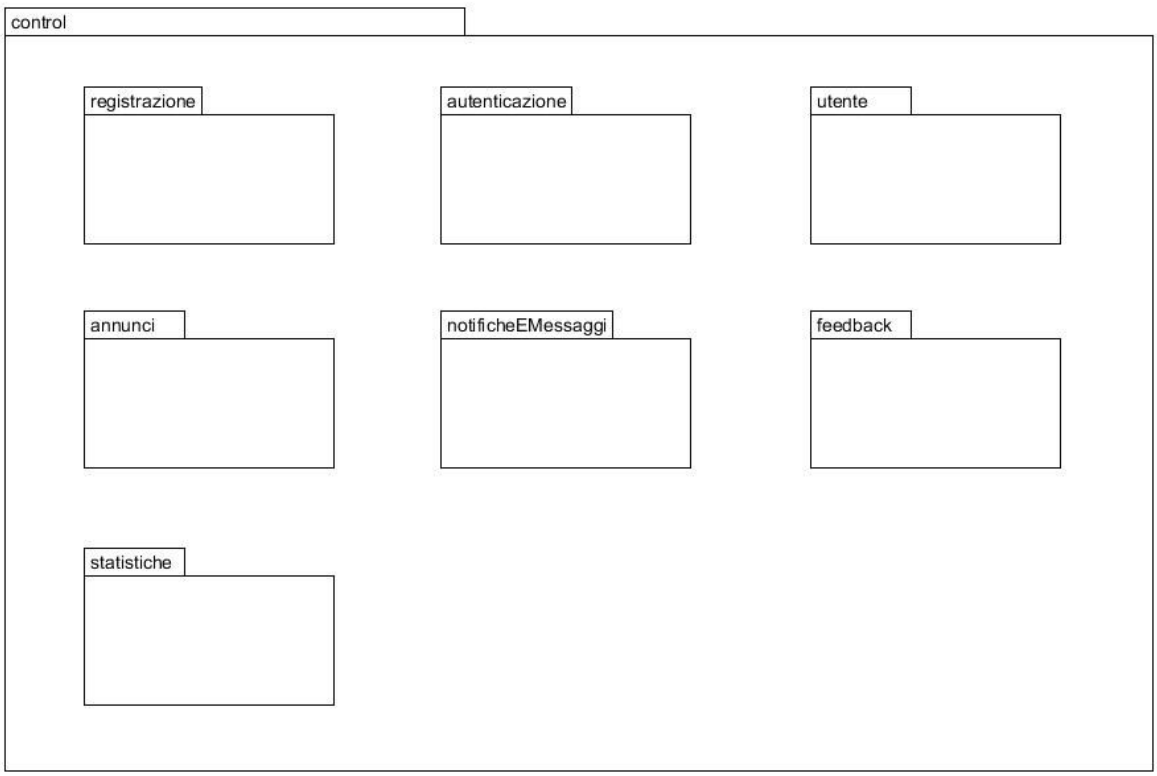
Classe:	Descrizione:
Annuncio.php	Descrive un Annuncio inserito nel sistema
Utente.php	Descrive un utente registrato al sistema
Commento.php	Descrive un commento inserito da un utente
Messaggio.php	Descrive un Messaggio inviato tra utenti
Macrocategoria.php	Descrive una Macrocategoria
Microcategoria.php	Descrive una Microcategoria
Candidatura.php	Descrive una candidatura ad un Annuncio
Feedback.php	Descrive una valutazione lasciata ad un utente

2.1.2 Packages Manager

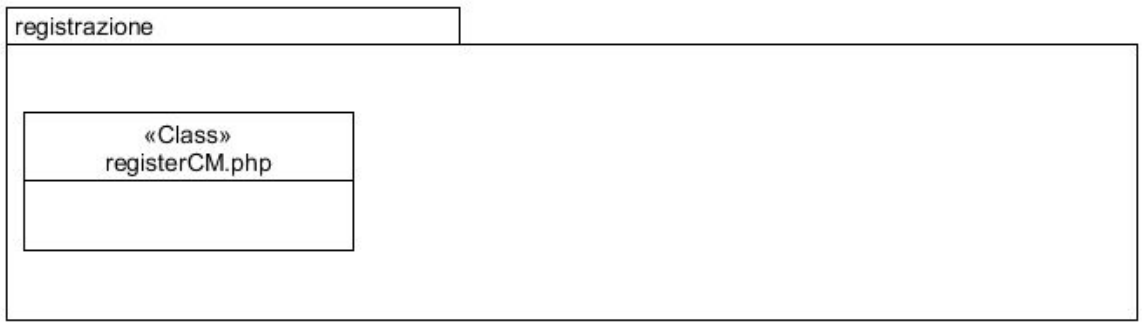


Classe:	Descrizione:
AnnuncioManager.php	Permette ad un utente di inserire, modificare, cancellare, ricercare e visualizzare un annuncio.
FeedbackManager.php	Permette la gestione dei feedback associati agli annunci e agli utenti.
MacroCategoriaManager.php	Permette la gestione delle MacroCatrgorie associati agli annunci e agli utenti.
Manager.php	Permette la gestione della connessione al database.
MessaggioManager.php	Permette la gestione dei messaggi scambiati tra due utenti del sistema.
MicroCategoriaManager.php	Permette la gestione delle MicroCatrgorie associati agli annunci e agli utenti.
NotificaManager.php	Permette la gestione delle notifiche del sistema, quindi come l'inoltro della notifica ad un utente.
UtenteManager.php	Permette di manipolare tutte le informazioni relative all'entità utente

2.1.3 Packages Control

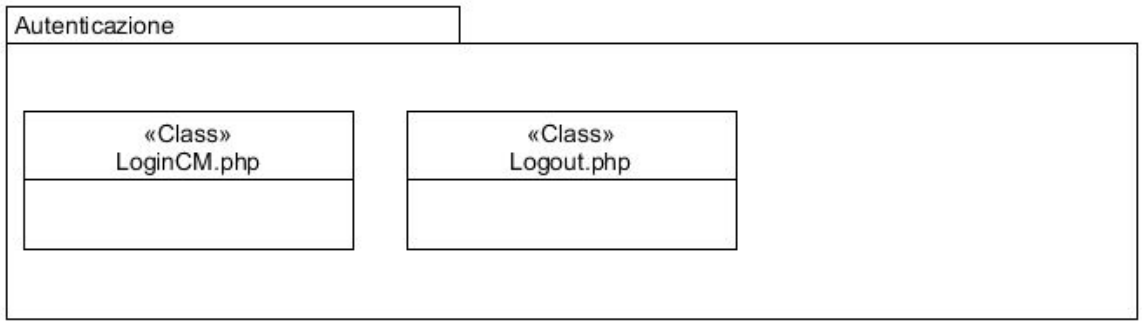


2.1.3.0 Registrazione



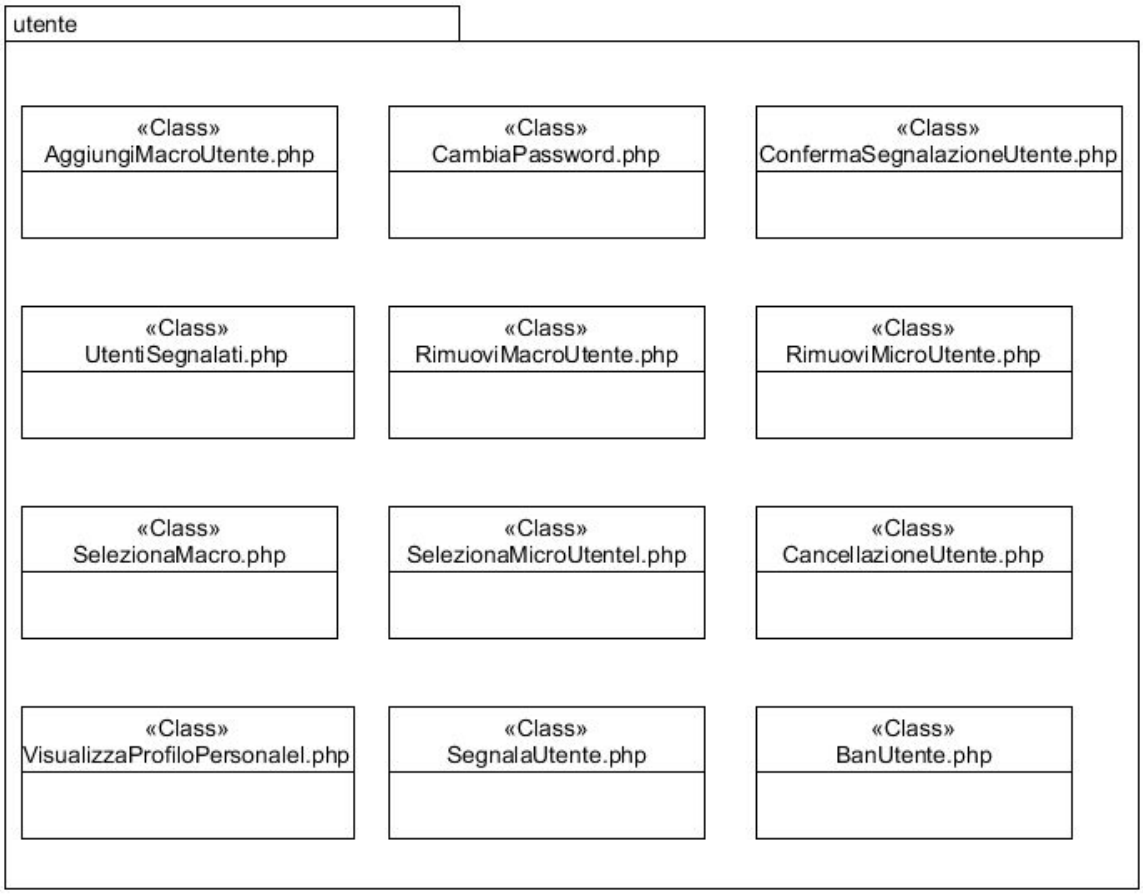
Classe:	**Descrizione:**
RegisterCM.php	Gestisce la registrazione di un utente.

2.1.3.1 Autenticazione



Classe:	**Descrizione:**
LoginCM.php	Effettua la Login dell'utente.
Logout.php	Effettua la Logout di un utente.

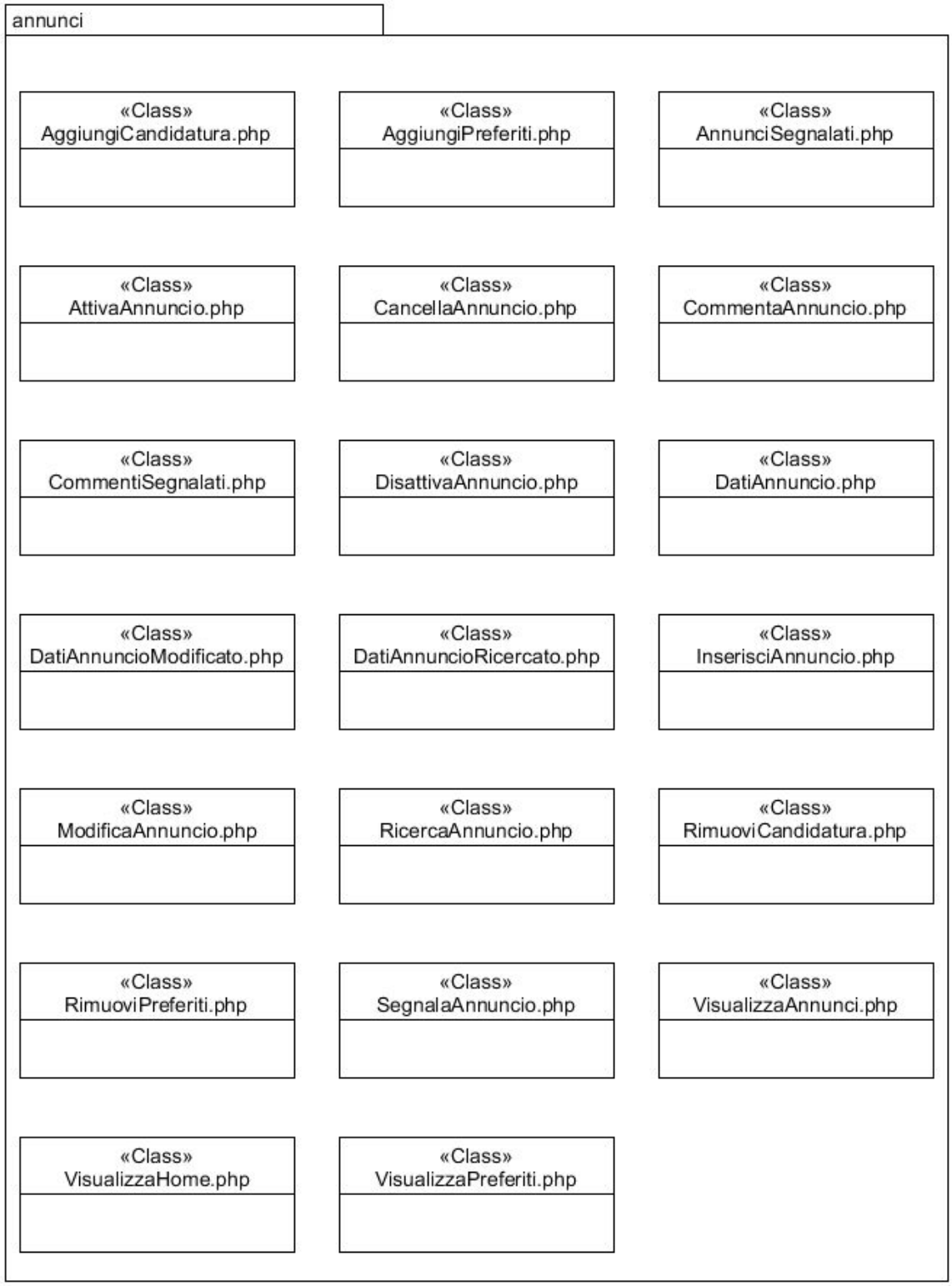
2.1.3.2 Utente



Classe:	**Descrizione:**
AggiungiMacroUtente.php	Permette di aggiungere una MacroCategoria alla propria area di competenza.
CambiaPassword.php	Permette di modificare la propria password.
ConfermaSegnalazioneUtente.php	Permette di confermare la segnalazione di un utente.
UtentiSegnalati.php	Ritorna la lista degli utenti segnalati.
RimuoviMacroUtente.php	Rimuove una MacroCategoria dalle competenze dell'utente.
RimuoviMicroUtente.php	Rimuove una microcategoria dalle competenze dell'utente.
SelezionaMacro.php	Carica la lista delle macro categorie e la lista delle macro categorie dell'utente.
SelezionaMicroUtentel.php	Carica la lista delle micro categorie e la lista delle macro categorie dell'utente.
CancellazioneUtente.php	Permette di cancellare un utente.
VisualizzaProfiloPersonalel.php	Permette di visualizzare la pagina profilo personale.

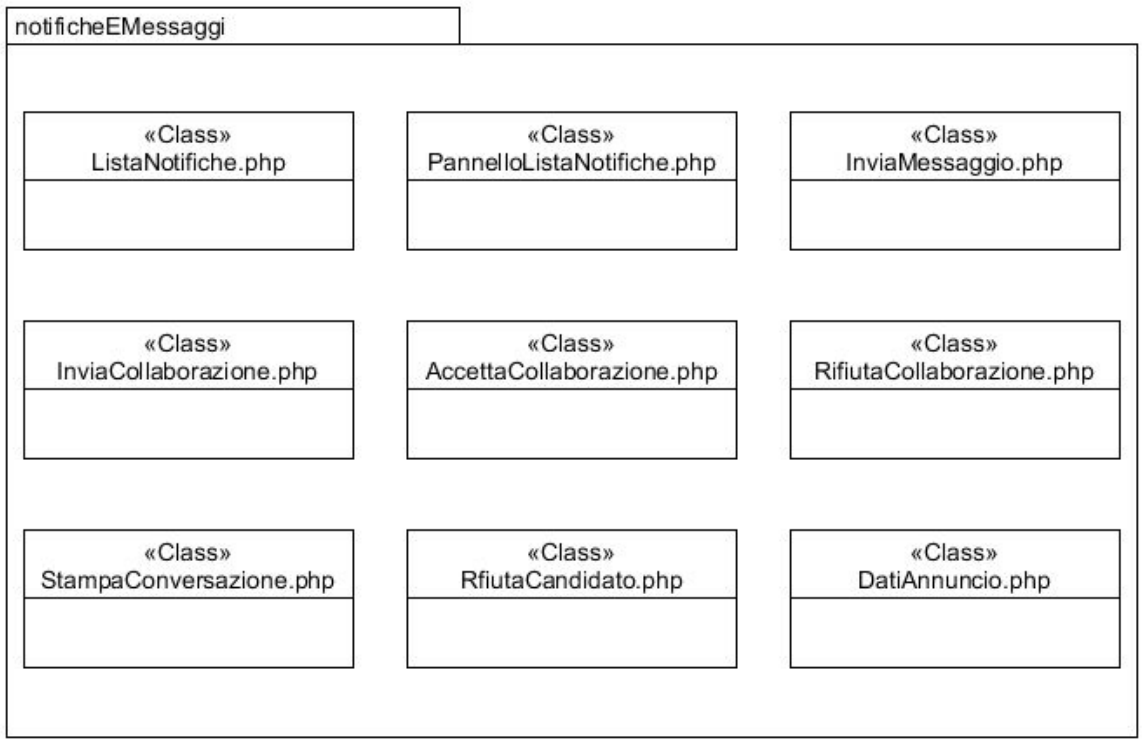
SegnalaUtente.php	Permette di segnalare un utente.
BanUtente.php	Permette di bannare un utente.
RiattivaUtente.php	Permette di riattivare un utente.

2.1.3.3 Annunci



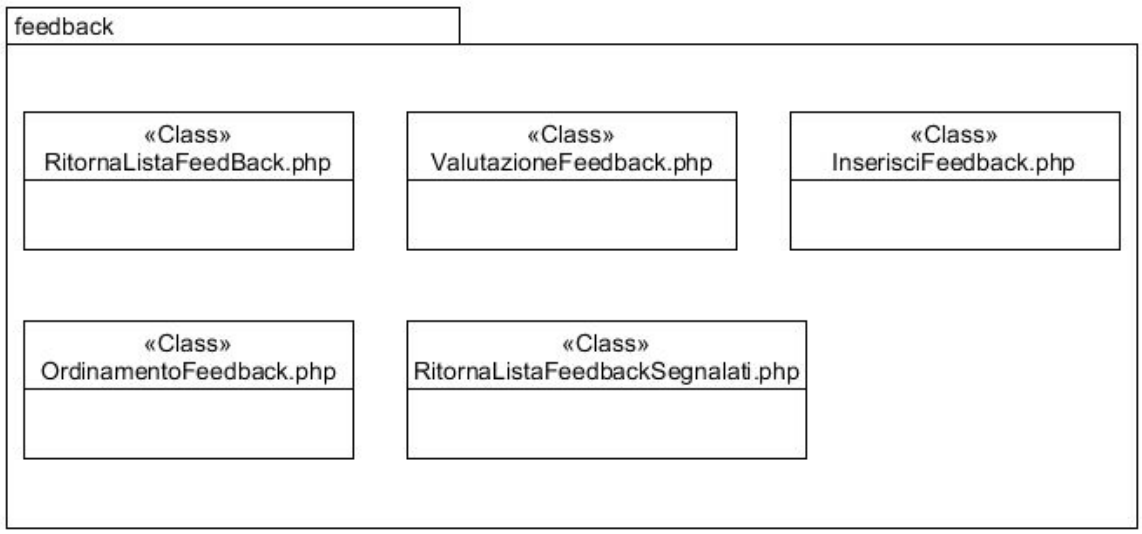
Classe:	**Descrizione:**
AggiungiCandidatura.php	Permette di aggiungere una candidatura all'annuncio.
AggiungiPreferiti.php	Permette di aggiungere un annuncio ai preferiti dell'utente.
AnnunciSegnalati.php	Permette di segnalare un annuncio.
AttivaAnnuncio.php	Permette di attivare un annuncio precedentemente disattivato.
CancellaAnnuncio.php	Permette di settare lo stato dell'annuncio in eliminato.
CommentaAnnuncio.php	Permette di lasciare un commento dato un determinato annuncio.
CommentiSegnalati.php	Permette di caricare la lista di tutti i commenti segnalati.
DisattivaAnnuncio.php	Permette di settare lo stato di un annuncio in disattivato.
DatiAnnuncio.php	Permette di controllare i parametri inseriti dall'utente quando inserisce un annuncio, e ne permette la creazione di un nuovo annuncio.
DatiAnnuncioModificato.php	Permette di controllare i parametri inseriti dall'utente quando modifica un annuncio proprietario.
DatiAnnuncioRicercato.php	Permette di controllare i parametri inseriti dall'utente quando ricerca un annuncio e chiama un metodo per ottenere la lista degli annunci con quei filtri settati.
InserisciAnnuncio.php	Permette di fare una redirect verso la view di inserimento degli annunci.
ModificaAnnuncio.php	Permette di fare redirect verso la view di modifica dell'annuncio, passando nella variabile di sessione, l'annuncio da modificare.
RicercaAnnuncio.php	Permette di fare redirect verso la view di ricercaAnnunci.
RimuoviCandidatura.php	Permette di rimuove una candidatura da un annuncio.
RimuoviPreferiti.php	Permette di rimuove un annuncio preferito dalla lista degli annunci preferiti di un utente.
SegnalaAnnuncio.php	Permette di segnalare un annuncio.
VisualizzaAnnunci.php	Permette di caricare la lista degli annunci appartenenti ad un utente, e inserisce la lista nella variabile di sessione.
VisualizzaHome.php	Permette di caricare la lista degli annunci da mostrare nella home page.
VisualizzaPreferiti.php	Permette di caricare la lista degli annunci preferiti di un utente e li passa alla view in una variabile di sessione.

2.1.3.4 Notifica e Messaggi



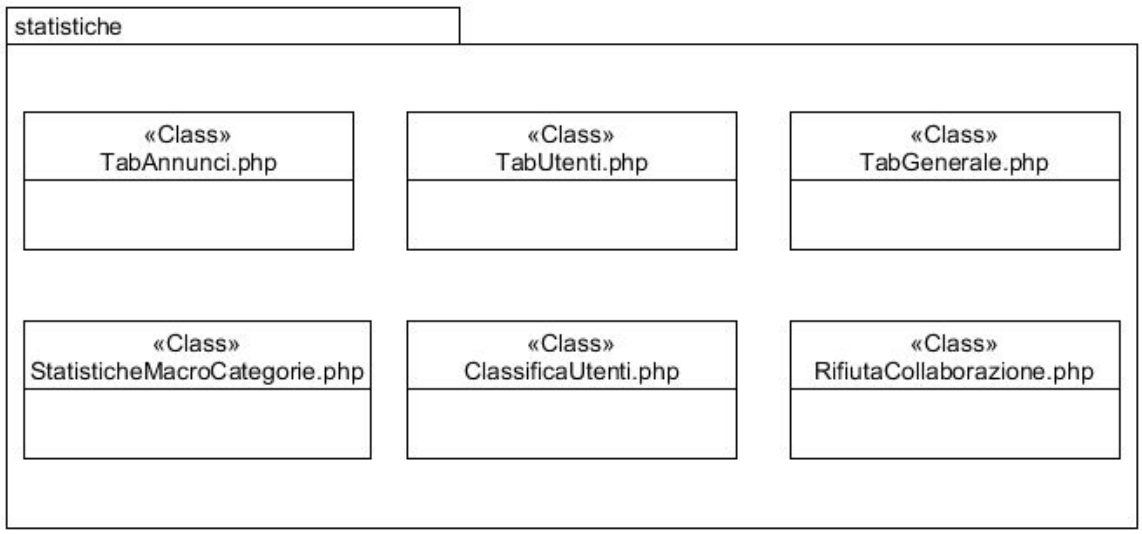
Classe:	**Descrizione:**
ListaNotifiche.php	Permette di caricare la lista delle notifiche totali relative ad un utente.
PannelloListaNotifiche.php	Permette di caricare le notifiche di un utente non lette all'interno del pannello delle notifiche del header.
InviaMessaggio.php	Invia un messaggio ad un altro utente.
InviaCollaborazione.php	Invia una collaborazione.
AccettaCollaborazione.php	Accetta la collaborazione.
RifiutaCollaborazione.php	Rifiuta una collaborazione.
StampaConversazione.php	Stampa la conversazione tra due utenti.
RfiutaCandidato.php	Rifiuta un Candidato

2.1.3.5 Feedback



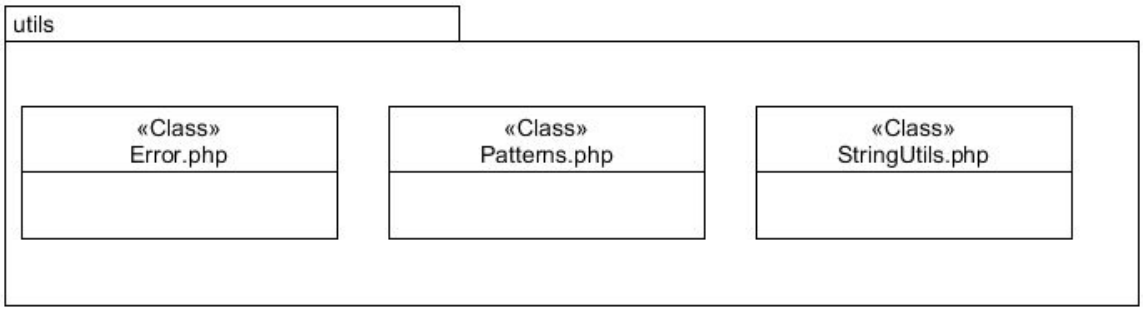
Classe:	**Descrizione:**
RitornaListaFeedBack.php	Ritorna la lista dei feedback relativi ad un utente.
ValutazioneFeedback.php	Permette al moderatore di valutare un Feedback.
InserisciFeedback.php	Permette all'utente di inserire un feedback.
OrdinamentoFeedback.php	Permette di caricare la lista dei feedback ordinata per data, nome utente e valutazione.
RitornaListaFeedbackSegnalati.php	Ritorna la lista dei feedback segnalati.

2.1.3.6 Statistiche



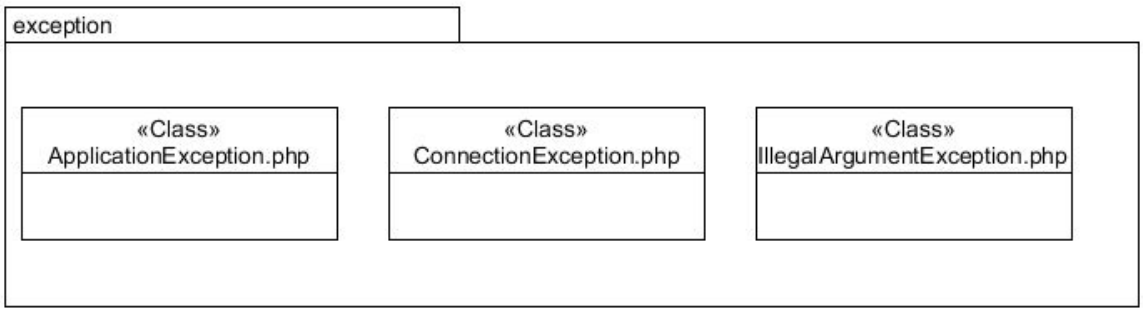
Classe:	**Descrizione:**
TabAnnunci.php	Permette di caricare i dati relativi agli annunci.
TabUtenti.php	Permette di caricare i dati d'interesse relativi agli utenti.
TabGenerale.php	Permette di caricare dati generali relativi agli annunci.
StatisticheMacroCategorie.php	Permette di caricare le statistiche relative a tutte le macro categorie.
ClassificaUtenti.php	Permette di caricare la classifica dei migliori utenti in base ai feedback.

2.1.4 Packages Utils



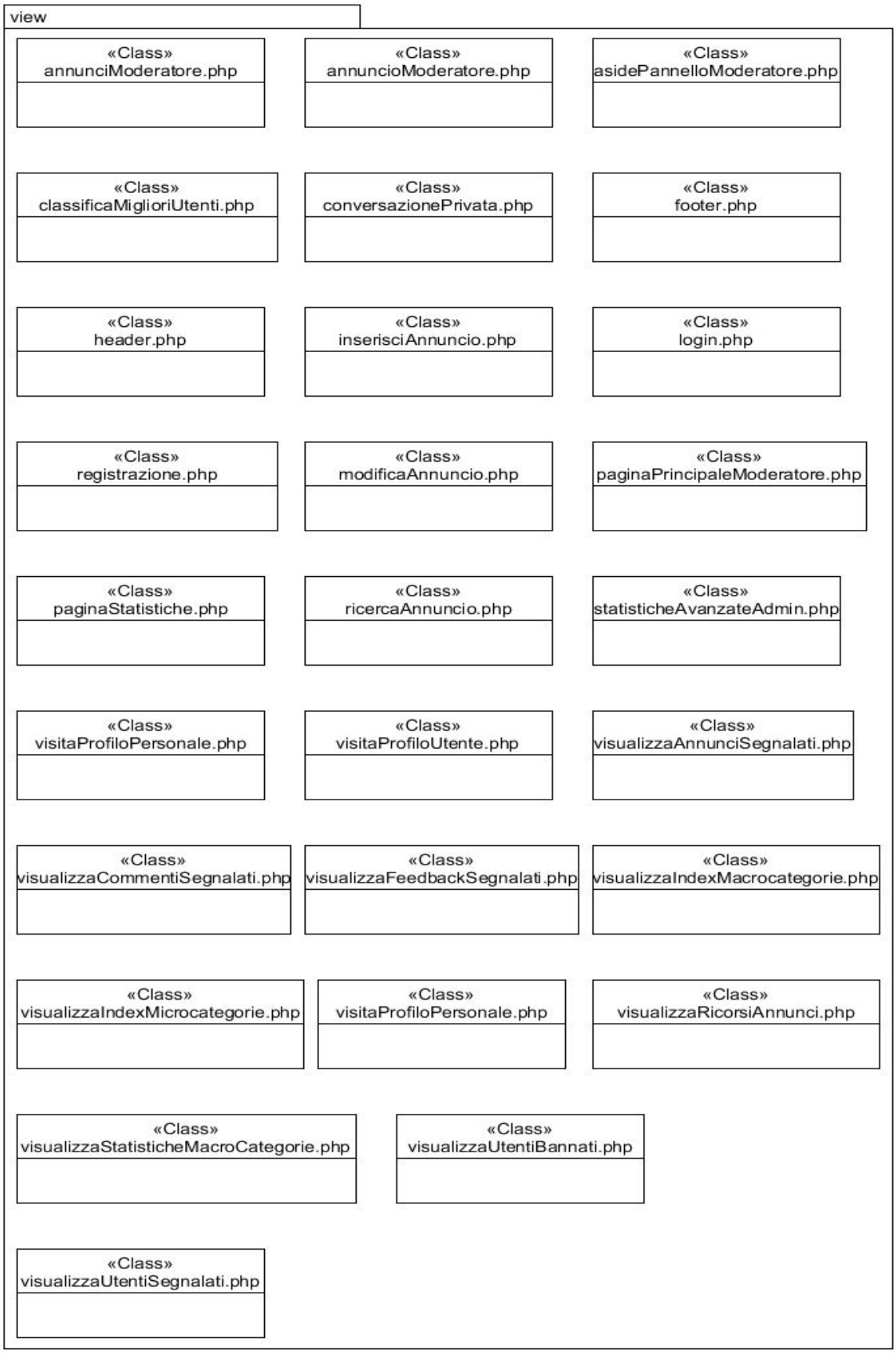
Classe:	**Descrizione:**
Error.php	La classe che contiene tutte le costanti stringhe utilizzate nei model per quanto riguarda le eccezioni.
Patterns.php	La classe che contiene tutte le espressioni regolari sottoforma di stringhe costanti.
StringUtils.php	La classe contiene dei metodi statici necessari per elaborare le stringhe

2.1.5 Packages Exception



Classe:	**Descrizione:**
ApplicationException.php	L'eccezione che viene lanciata quando non viene eseguita una query.
ConnectionException.php	L'eccezione che viene lanciata quando non c'è connessione al database.
IllegalArgumentException.php	L'eccezione che viene lanciata quando i parametri passati ad un metodo non sono corretti.

2.1.6 Packages View



Classe:	**Descrizione:**
annunciModeratore.php	Visualizza la pagina del moderatore relativa agli annunci.
annuncioModeratore.php	Visualizza la pagina di un annuncio dalla vista del moderatore.
asidePannelloModeratore.php	Visualizza la barra
classificaMiglioriUtenti.php	Viusalizza la pagina delle statistiche con la calssifica dei migliori utenti.
conversazionePrivata.php	Visualizza la sezione della chat relativa ad una conversazione privata tra due utenti
footer.php	Visualizza il footer di Crowdmine
header.php	Visualizza l'header di Crowdmine.
inserisciAnnuncio.php	Visualizza la pagina che permette di inserire un nuovo annuncio.
login.php	Visualizza la pagina che permette di effettuare l'autenticazione.
registrazione.php	Visualizza la pagina che permette di iscriversi al sistema.
modificaAnnuncio.php	Visualizza la pagina di modifica di un annuncio.
paginaPrincipaleModeratore.php	Visualizza la pagina dove il moderatore può gestire il sistema.
paginaStatistiche.php	Visualizza la pagina principale delle statistiche
ricercaAnnuncio.php	Pagina di visualizzazione di un annuncio.
statisticheAvanzateAdmin.php	Visualizza la pagina principale della gestione delle statistiche avanzate.
visitaProfiloPersonale.php	Visualizza la pagina del proprio profilo.
visitaProfiloUtente.php	Visualizza la pagina del profilo di un altro utente.
visualizzaAnnunciSegnalati.php	Visualizza la pagina dove è presente la lista degli annunci segnalati.
visualizzaCommentiSegnalati.php	Visualizza la pagina dove è presente la lista dei commenti segnalati.
visualizzaFeedbackSegnalati.php	Visualizza la pagina dove è presente la lista dei feedback segnalati.
visualizzaIndexMacrocategorie.php	Visualizza la pagina dove è presente la lista di tutte le macro categorie presenti nel sistema.
visualizzaIndexMicrocategorie.php	Visualizza la pagina dove è presente la lista di tutte le micro categorie presenti nel sistema.
visualizzaRicorsiAlBan.php	Visualizza la pagina dove è presente la lista degli utenti che

	hanno effettuato il ricorso al ban.
visualizzaRicorsiAnnunci.php	Visualizza la pagina dove è presente la lista degli annunci segnalati i quali autori hanno fatto ricorso.
visualizzaStatisticheMacroCategorie.php	Visualizza la pagina dove è presente un grafico che descrive l'andamento delle macro categorie del sistema.
visualizzaUtentiBannati.php	Visualizza la pagina dove è presente la lista degli utenti bannati.
visualizzaUtentiSegnalati.php	Visualizza la pagina dove è presente la lista degli utenti segnalati.

2.1.7 Package Filter

Classe	Descrizione
Filter.php	
FilterUtils.php	
SearchByDateInterval.php	
SearchByidFilter.php	
SearchByStatus.php	
SearchByTitleFilter.php	
SearchByUserIdFilter.php	

3. Interfaccia delle Classi

3.1 Packages Manager

3.1.0 Annuncio Manager

Si tratta del Manager che gestisce il Model dell'entity Annuncio.

- function createAnnuncio($userid, $date, $title, $location, $microcat, $remuneration, $type, $description)
  Questo metodo permette di creare un oggetto Annuncio.

- function addMicrosToAnnuncio($id, $microcat)
  Questo metodo permette di aggiungere una micro categoria ad un determinato annuncio.

- function updateAnnuncio($id, $userid, $date, $title, $location, $microcat = null, $remuneration, $type, $description)
  Questo metodo permette di aggiornare un annuncio all'interno del database.

- function deleteAnnuncio($idAnnuncio)
  Questo metodo permette di eliminare un annuncio all'interno del database.

- function searchAnnuncio($filters)
  Questo metodo restituisce una lista di annunci che rispettano il criterio di ricerca passato come parametro.

- function getAnnuncio($id)
  Questo metodo restituisce l'annuncio con l'id specificato come parametro.

- function searchAnnunciUtente($idUtente)
  Questo metodo restituisce una lista di annunci che appartengono ad un determinato utente.

- function addCandidatura($idAnnuncio, $idUtente, $message, $data)
  Questo metodo permette di aggiungere una candidatura ad un annuncio.

- function addToFavorites($idAnnuncio, $idUtente, $data)
  Questo metodo permette di aggiungere un annuncio dai preferiti di un utente.

- function removeFromFavorites($idAnnuncio, $idUtente)
  Questo metodo permette di rimuovere un annuncio dai preferiti di un utente.

- function getAnnunciHomePage()
  Questo metodo restituisce una lista di annunci che verranno visualizzati nella home page.

- function reportAnnuncio($idAnnuncio)
  Questo metodo permette di segnalare un annuncio.

- function getReportedAnnunci()
  Questo metodo restituisce la lista degli annunci segnalati

- function getAnnuncioWithCandidati($idAnnuncio)
  Questo metodo restituisce la lista di tutti i candidati ad una lista.

- function commentAnnuncio($idAnnuncio, $idUtente, $comment, $date)
  Questo metodo permette di aggiungere un commento ad un annuncio.

- function updateStatus($idAnnuncio, $newStatus, $oldStatus = null)
  Questo metodo permette di modificare lo stato di un annuncio soltanto se il suo vecchio stato è nullo.

- function validateAnnuncio($idAnnuncio)
  Questo metodo permette di modificare lo stato di un annuncio ad attivo soltanto se il suo stato precedente è revisione.Questo metodo può essere chiamato soltanto da un moderatore.

- function confirmValidationAnnuncio($idAnnuncio)
  Questo metodo permette di modificare lo stato di un annuncio ad attivo soltanto se il suo stato precedente è segnalato.

- function sendClaim($idAnnuncio, $message)
  Questo metodo permette di modificare lo stato di un annuncio in ricorso soltanto se il suo stato precedente è disattivato.

- function sendSuspension($idAnnuncio, $message = "")
  Questo metodo permette di modificare lo stato di un annuncio in disattivato.

- function sendConfirmation($idAnnuncio)
  Questo metodo permette di modificare lo stato di un annuncio ad attivo soltanto se il suo stato precedente è amministratore.Questo metodo può essere chiamato soltanto da un amministratore.

- function sendToAdmin($idAnnuncio)
  Questo metodo permette di modificare lo stato di un annuncio in amministratore.

- function getClaimedAnnunciList()
  Questo metodo restituise la lista di tutti gli annunci con lo stato di ricorso.

- function getAnnunciCount($filters)
  Questo metodo restituisce il numero di annunci che corrispondono ai criteri specificati.

.

- function getAnnunciRanking($filters = [])
  Questo metodo restituisce la classifica dei migliori annunci

3.1.1 Utente Manager

Si tratta del Manager che gestisce il Model dell'entity Utente

- public function createUser($id, $nome, $cognome,$descrizione, $telefono, $dataNascita, $citta, $email, $password, $stato, $ruolo, $immagineProfilo)
  Crea un Utente non persistente.

- public function updateUtente($user)
  Aggiorna un Utente all'interno del database.

- public function findUtenteById($userId)
  Ricerca l'utente con l'id specificato.

- private function findUtenteByUserName($nome, $cognome)
  Ricerca gli utenti tramite la user name.

- public function findUserByMicroAvgOverRatio($microCategoria, $numStelle)
  Ricerca gli utenti con un rating maggiore o uguale di quello passato.

- public function findAll()
  Ritorna la lista di tutti gli utenti.

- public function getReportedUtente()
  Ritorna la lista degli utenti Segnalati.

- public function getBannedUtente()
  Ritorna la lista degli utenti Bannati.

- public function getAppealUtente()
  Ritorna la lista degli utenti che hanno fatto Ricorso.

- public function checkEmail($email)
  Controlla se esiste un utente con la mail passata.

- public function checkPassword($userId, $password)
  Controlla se esiste un utente con la password passata.

- public function addMicroCategoria($user,$microcategoria)
  Aggiunge una microcategoria alle competenze di un Utente.

- public function removeMicroCategoria($user, $microcategoria)
  Rimuove una microcategoria alle competenze di un Utente.

- public function login($email, $password)
  Effettua il login di un Utente.

- public function register($user)
  Effettua la registrazione di un utente, inserendolo nel database.

- public function searchUtente($nome, $cognome)
  Cerca un utente per nome.

3.1.2 Notifica Manager

- public function createNotifica($id, $data, $tipo, $info, $letto)
  Questo metodo crea una Notifica non persistente.

- public function insertNotifica($id, $data, $tipo, $info, $letto)
  Questo metodo crea una Notifica persistente.

- public function sendToDispatcher($listaUtenti, $idNotifica)
  Questo metodo invia una notifica al dispatcher.

- public function getNotifica($idUtente)
  Questo metodo ritorna la lista di tutte le notifiche di un utente.

- public function getNotificaNotVisualized($idUtente)
  Questo metodo ritorna la lista di notifiche di un utente non ancora lette.

- public function loadFromDispatcher($idUtente)
  Questo metodo carica le notifiche di un utente dal dispatcher.

3.1.3 Feedback Manager

- function insertFeedback($id=null,$idUtente,$idAnnuncio,$idValutato,
  $valutazione,$corpo,$data,$stato,$titolo)
  Questo metodo permette l'inserimento di un feedback nel database.

- function createFeedback($id=null,$idUtente,$idAnnuncio,$idValutato,
  $valutazione,$corpo,$data,$stato,$titolo)
  Questo metodo permette di creare un oggetto Feedback.

- function checkCollaboration($idVotante,$idAnnuncioVotato)
  Questo metodo permette di controllare se fra due utenti è avvenuta o meno una collaborazione.

- function getFeedbackById($id)
  Questo metodo permette di ottenere un feedback utilizzando l'id come chiave di ricerca.

- function setStatus($id,$stato)
  Questo metodo permette di settare lo stato ad un oggetto Feedback.

- function getListaFeedback($idUtente)
  Questo metodo permette di ottenere la lista dei feedback lasciati ad un utente.

- function getListaFeedbackByMicrocategoria($idUtente, $microCategoria)
  Questo metodo permette di ottenere la lista dei feedback relativi ad un certo utente e una micro categoria.

- function sortListaFeedback($idUtente,$param)
  Questo metodo permette di ordinare la lista dei feedback lasciati ad un utente.

- function getFeedbackAdmin()
  Questo metodo permette di ottenere la lista dei feedback che hanno lo stato settato ad amministratore.

- function getFeedbackSegnalati()
  Questo metodo permette di ottenere la lista dei feedback che hanno lo stato settato a segnalato.

- function removeFeedback($idFeedback)
  Questo metodo setta lo stato di un feedback a eliminato.

3.1.4 MicroCategoria Manager

- function createMicrocategoria($id, $nomeMicro, $idMacro)
  Questo metodo permette di creare un oggetto Micro Categoria non persistente.

- function addMicrocategoria($microcategoria)
  Questo metodo permette di aggiungere un oggetto Micro Categoria al database.

- function deleteMicrocategoria($microcategoria)
  Questo metodo permette di cancellare un oggetto Micro Categoria.

- function editMicrocategoria($microcategoria)
  Questo metodo permette di modificare un oggetto Micro Categoria.

- function checkMicrocategoria($nome)
  Questo metodo controlla l'esistenza di un oggetto Micro Categoria

- function findMicrocategoriaById($idMicro)
  Questo metodo restituisce un oggetto Micro Categoria in base all'id;

- function findMicrocategoriaByNome($nome)
  Questo metodo restituisce ricerca un Microcategoria per nome.

- function findAll()
  Questo metodo ritorna la lista di tutte le Micro Categorie inserite nel sistema.

- function findBestMicrocategoria($macrocategoria)
  Questo metodo restituisce la lista di oggetti Micro Categorie secondo in base alle competenze scelte degli utenti.

3.1.5 MacroCategoria Manager

- function createMacrocategoria($nome)
  Questo metodo permette di creare una Macro Categoria non persistente.

- function getMacroByName($nome)
  Questo metodo restituisce un oggetto Macro Categoria in base al nome.

- function findListMacorocategoria()
  Questo metodo restituisce la lista di oggetti Macro Categoria in base alla competenze scelte dagli utenti.

3.1.6 Messaggio Manager

- function sendMessaggio($id, $corpo, $letto, $data, $idMittente, $idDestinatario)
  Questo metodo permette di inviare un oggetto messaggio a un altro utente.

- function loadMessaggi($idUtente)
  Questo metodo carica i messaggi relativi all'Utente.

- function loadMessaggio($idMittente, $idDestinatario)
  Questo metodo carica la conversazione tra due utenti.

- function sendRichiestaCollaborazione($id, $idMittente, $idAnnuncio, $corpo, $data_risposta, $data_inviata, $richiesta_inviata, $richiesta_accettata)
  Questo metodo invia una richiesta di collaborazione ad un altro utente.

- function agreeCollaborazione($idAnnuncio)
  Questo metodo conferma la collaborazione tra due Utenti su un Annuncio rilasciato.

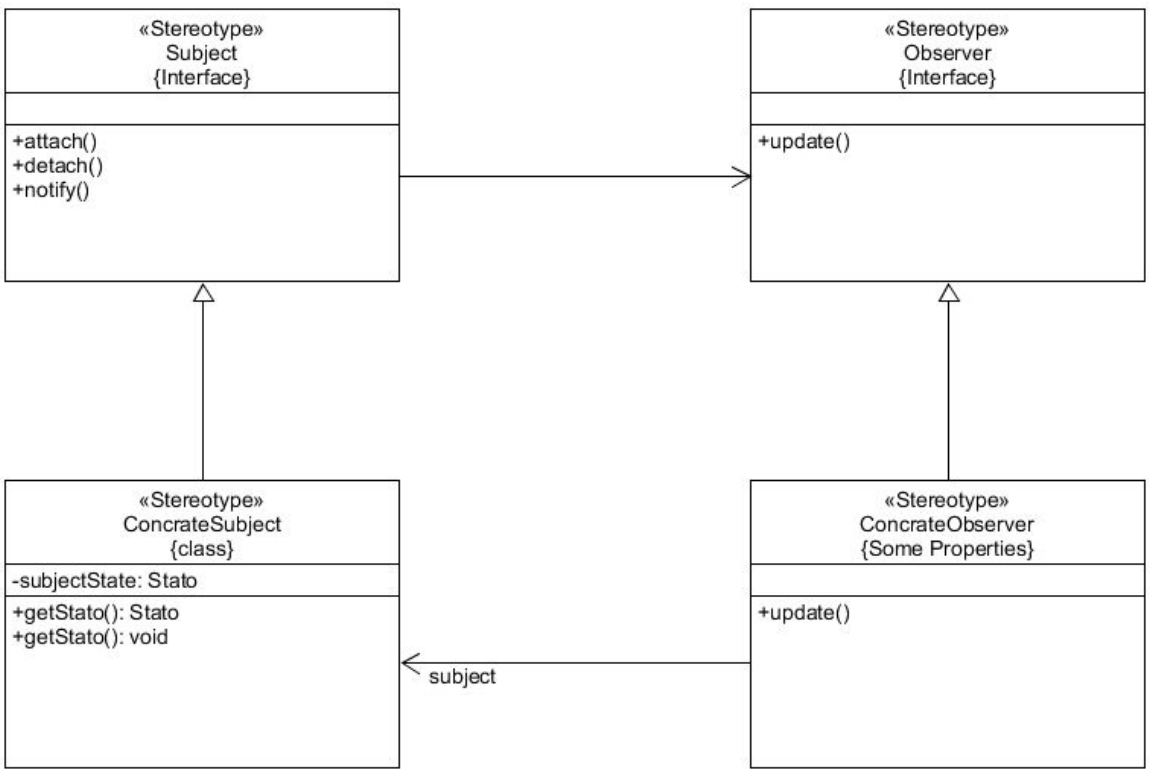
4. Design Patterns

Per la gestione delle notifiche è stato utilizzato il pattern Observer, questo pattern è di tipo comportamentale.
Il pattern si basa su due tipi di oggetti, i Subject che sono oggetti che possono generare un certo tipo di evento che ne cambia lo stato, gli Observer invece sono quegli oggetti che una volta registrati in una coda, relativa ad un certo Subject di interesse, sono avvisati di un'eventuale sua modifica.

In questo caso per l'implementazione si è usata la struttura del pattern Observer offerta dal pacchetto SPL del linguaggio PHP (disponibile a partire da PHP 5 e successive versioni), questo pacchetto dispone di una serie di interfacce utili.
Nel caso di questo progetto vengono usate due interfacce: **splSubject** e **splObserver**;
alla prima fanno riferimento tre metodi:

1. SplSubject::attach: accetta e attribuisce un observer memorizzandolo in un oggetto, in pratica associa degli observers ad un subject;
2. SplSubject::detach: libera l'oggetto in cui l'observer è archiviato da quest'ultimo, in pratica rimuove l'associazione tra subject ed observers;
3. SplSubject::notify: effettua una notifica a tutti gli observers attribuiti relativamente ad un evento che riguarda il subject.

Mentre ad splObserver è associato il solo metodo spl::update() che notifica i cambiamenti di un Subject.
Di seguito il grafico del pattern Observer generico:



Un esempio di grafico che descrive il nostro sistema, invece, è riportato di seguito: