

ODD Object Design Document

DInfBoard

|  |  |
| --- | --- |
| Riferimento |  |
| Versione | 1.0 |
| Data | 16/12/2018 |
| Destinatario | Filomena Ferrucci – Carmine Gravino |
| Presentato Da | Daniele Cerardi – Antonio Jr, Citarella – Francesco Di Lascio – Simone Pirozzi – Antonio Vitale |
| Approvato Da |  |

|  |  |
| --- | --- |
| Nome | Matricola |
| Daniele Cerardi | 0512104555 |
| Antonio Junior Citarella | 0512104543 |
| Francesco Di Lascio | 0512103219 |
| Simone Pirozzi | 051214723 |
| Antonio Vitale | 0512104819 |

|  |  |  |  |
| --- | --- | --- | --- |
| Data | Versione | Cambiamenti | Autori |
| 06/12/2018 | 0.1 | Sistemazione Template Documento | Antonio Junior Citarella |
| 09/12/2018 | 0.2 | Individuazione Trade-Off e Pattern da utilizzare | Antonio Junior Citarella  Antonio Vitale |
| 10/12/2018 | 0.2.1 | Stesura dell’introduzione e dei diagrammi di pattern | Antonio Junior Citarella  Antonio Vitale |
| 11/12/2018 | 0.2.2 | Diagramma dei Package e prima stesura Interfacce delle classi e contratti | Antonio Junior Citarella |
| 12/12/2018 | 0.2.3 | Class Diagram e Ridefinizione Interfacce e contratti | Antonio Junior Citarella |
| 13/12/2018 | 0.2.4 | Stesura Tabella Classi con descrizione e Diagramma dei Pattern | Antonio Junior Citarella  Antonio Vitale |
| 14/12/2018 | 0.2.5 | Inserimento Pattern Definitivi e Class Diagram finale | Antonio Vitale |
| 16/12/2018 | 0.2.6 | Inserimento ulteriori Interfacce e ulteriori contratti delle classi | Antonio Junior Citarella  Francesco Di Lascio |
| 09/01/2019 | 0.2.7 | Modifiche e controlli sulle classi. Revisione complessiva. | Daniele Cerardi |

Sommario

1. [1. Introduzione 3](#_Toc532649507)

[1.1. Trade-off dell’Object Design](#_Toc532649508)

[1.1.1 Trade-Off generali.](#_Toc532649509)

[1.1.2 Linee Guida e Convenzioni.](#_Toc532649510)

[1.1.3 Componenti Off-The-Shelf](#_Toc532649511)

[1.1.4 Overview del Documento](#_Toc532649512)

[1.2. Linee guida per la documentazione dell’interfaccia](#_Toc532649513)

[1.3. Definizioni, acronimi e abbreviazioni](#_Toc532649514)

[1.4. Riferimenti](#_Toc532649515)

1. [2. Pattern 6](#_Toc532649516)
2. [3. Packages 8](#_Toc532649517)
3. [4. Interfacce delle classi 10](#_Toc532649518)

# Introduzione

## Trade-off dell’Object Design

## 1.1.1 Trade-Off generali.

### Tempo di Consegna vs Funzionalità

Si è preferito effettuare una consegna rapida presentando quanto prima il prodotto al cliente piuttosto che arricchire la piattaforma con funzionalità ulteriori; In particolare, saranno disponibili all’utente quelle specificate dal cliente nei documenti di contratto stipulati con esso.

### Tempo di Documentazione vs Interfaccia

Produrremo una documentazione completa e dettagliata di come il sistema dovrà essere utilizzato, realizzando un’interfaccia meno curata esteticamente, in quanto possibile oggetto di miglioramento per rilasci futuri del sistema.

### Response Time vs Memoria

Conoscendo le dimensioni dello spazio da riservare alla piattaforma, si è preferito avere una memoria limitata all’utilizzo di un database presente su un solo Web Server con lo scopo, però, di rendere veloci le operazioni effettuate dagli utenti sul sistema.

## Componenti Off-The-Shelf

Le componenti off-the-shelf che utilizzeremo sono composte principalmente da librerie JEE e JavaScript, attraverso le quali sarà possibile realizzare operazioni complesse con poche linee di codice. La prima libreria JEE che utilizzeremo è la javax.mail, che ci permetterà di inviare e-mail agli utenti nel tentativo di recuperare le credenziali d’accesso alla piattaforma. La seconda libreria che utilizzeremo è la javax.json, la quale ci fornirà un parser JSON per effettuare operazioni sul database in maniera asincrona attraverso AJAX, uno strumento messoci a disposizione da JavaScript. L’ultima componente utilizzata sarà la libreria JQuery, inserita all’interno delle nostre pagine html per essere utilizzata dal nostro codice Javascript al fine di effettuare operazioni complesse con poche righe di codice.

## Overview del Documento

Segue nel documento la lista delle linee guida da seguire nel processo d’implementazione. La seconda sezione sarà dedicata ai Pattern di design scelti per la piattaforma e alle informazioni sulla loro struttura. La terza sezione, invece, descriverà la lista dei pacchetti implementativi scelti in base alle convenzioni appena citate. La quarta sezione, infine, si occuperà di descrivere l’insieme d’interfacce che si intende utilizzare durante la scrittura del codice.

## Linee guida per la documentazione dell’interfaccia

* La convenzione che deve essere adottata da tutti i team member per quanto riguarda i nomi delle variabili, è la nota Camel Notation.

**Organizzazione dei file**

* + Ogni file:
* Sarà sviluppato e inserito all’interno del modulo a cui appartiene. Ogni pagina della piattaforma (login, inserimento file, visualizzazione indicatori, etc.) deve essere implementata in file separati;
* Sarà diviso in più file, se raggiunge una lunghezza tale da divenire difficile da leggere e comprendere;
* La convenzione per quanto riguarda i nomi dei file, delle operazioni e delle variabili è quella di avere nomi evocativi, ma soprattutto in lingua italiana. Le uniche eccezioni sono: GET e SET. I termini GET e SET sono stati inseriti nella nostra terminologia data la familiarità con il linguaggio di programmazione JAVA e soprattutto perché anche la migliore traduzione in italiano non regge il confronto.
* Organizzare in una cartella i file delle librerie usate e le altre risorse scaricate necessarie per lo sviluppo del progetto.

**Spostamento di linee**

* Quando un’espressione supera la lunghezza della linea, occorre spezzarla secondo i seguenti principi generali:
  + - Interrompere la linea dopo una virgola;
    - Interrompere la linea prima di un operatore;
    - Preferire interruzioni di alto livello rispetto ad interruzioni di basso livello (interrompere laddove non si interrompe un discorso logico, discorso valido soprattutto per le formule es. (3+4) \* 2 interrompere prima della moltiplicazione senza spezzare gli operandi in parentesi);
    - Allineare la nuova linea con l’inizio dell’espressione nella linea precedente;
    - Se le regole precedenti rendono il codice più confuso o il codice è troppo spostato verso il margine destro, utilizzare solo otto spazi di indentazione.

**Indentazione**

* L’indentazione deve essere effettuata con un TAB e qualunque sia il linguaggio usato per la produzione di codice, ogni istruzione deve essere opportunamente indentata.

*Es.*

<html>

<head>

</head>

<body>

</body>

</html>

Deve essere sostituita da:

<html>

<head>

</head>

<body>

</body>

</html>

Questa pratica deve essere usata soprattutto per le istruzioni FOR, IF.

È buona pratica scendere di livello.

**Inizializzazione**

Inizializzare le variabili locali nel punto in cui sono state dichiarate a meno che il suo valore iniziale non dipenda da un calcolo che occorre eseguire prima.

**Posizione**

Mettere le dichiarazioni all’inizio dei blocchi. Non aspettare di dichiarare le variabili al loro primo uso: può confondere il programmatore inesperto e impedire la portabilità del codice dentro lo scope. L’unica eccezione a questa regola sono gli indici dei cicli for che in Java possono essere dichiarati nell’ istruzione stessa. Evitare dichiarazioni locali che nascondono dichiarazioni a più alto livello. Ad esempio, non dichiarare una variabile con lo stesso nome in un blocco interno.

**Parentesi**

Ogni tag di apertura deve essere necessariamente seguito dall’apposito tag di chiusura (eccetto i tag self-closing).

Una convenzione importante, per quanto riguarda l’inserimento di numeri o di valori costanti, è quella di non usare una codifica fissa (hard coding), ma di associare sempre il valore ad una variabile o semplicemente definire una macro-variabile che può essere richiamata da eventi ed essere parametrizzata. In questo modo si facilita la modifica, sostituendo solo il valore della variabile o macro, in un unico posto.

## Definizioni, acronimi e abbreviazioni

* **DBMS:** Data Base Management System.
* **Off-The-Shelf:** Servizi esterni di cui viene fatto utilizzo da terzi.
* **HTML:** Linguaggio di mark-up per pagine web.
* **CSS:** Linguaggio usato per definire la formattazione di pagine web.
* **JavaScript:**  [Linguaggio di scripting](https://it.wikipedia.org/wiki/Linguaggio_di_scripting) [orientato agli oggetti](https://it.wikipedia.org/wiki/Programmazione_orientata_agli_oggetti) e agli [eventi](https://it.wikipedia.org/wiki/Programmazione_orientata_agli_eventi), comunemente utilizzato nella [programmazione Web](https://it.wikipedia.org/wiki/Programmazione_Web) [lato client](https://it.wikipedia.org/wiki/Lato_client) per la creazione, in [siti web](https://it.wikipedia.org/wiki/Sito_web) e [applicazioni web](https://it.wikipedia.org/wiki/Applicazione_web), di effetti dinamici [interattivi](https://it.wikipedia.org/wiki/Interattivit%C3%A0)tramite [funzioni](https://it.wikipedia.org/wiki/Funzione_(informatica)) di [script](https://it.wikipedia.org/wiki/Script) invocate da *eventi* innescati a loro volta in vari modi dall'utente sulla [pagina web](https://it.wikipedia.org/wiki/Pagina_web) in uso
* **Parser:** Processo che analizza un flusso di dati e ne determina la struttura.
* **JSON:** E’ un formato adatto per lo scambio dei dati in applicazioni client-server.
* **Camel Notation:** Consiste nello scrivere più parole insieme delimitando la fine e l’inizio di una nuova parola con una lettera maiuscola.
* **JAVAX.MAIL:** Libreria Java utilizzata per la gestione di posta elettronica tramite codice.
* **JAVAX.JSON:** Libreria Java utilizzata per il parsing di oggetti JSON.

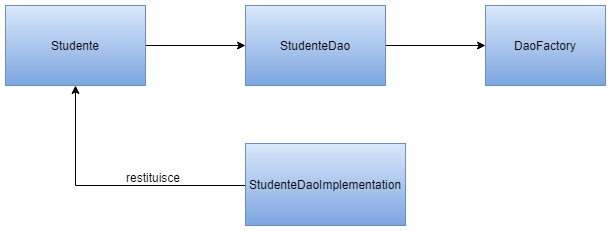
## Riferimenti

* Bernd Bruegge & Allen H. Dutoit, *Object-Oriented Software Engineering: Using UML, Patterns and Java*, (2nd edition),Prentice-Hall, 2003
* Ian Sommerville, Software Engineering, Addison Wesely
* SDD\_DDI\_1.0
* <https://github.com/SheetJS/js-xlsx>

**Antonio Junior Citarella**

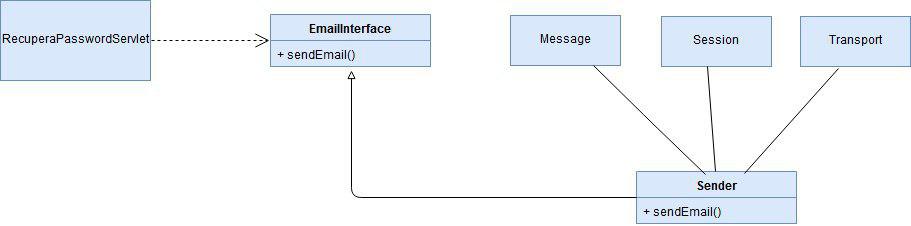
# Pattern

PATTERN DAO



**Antonio Vitale**

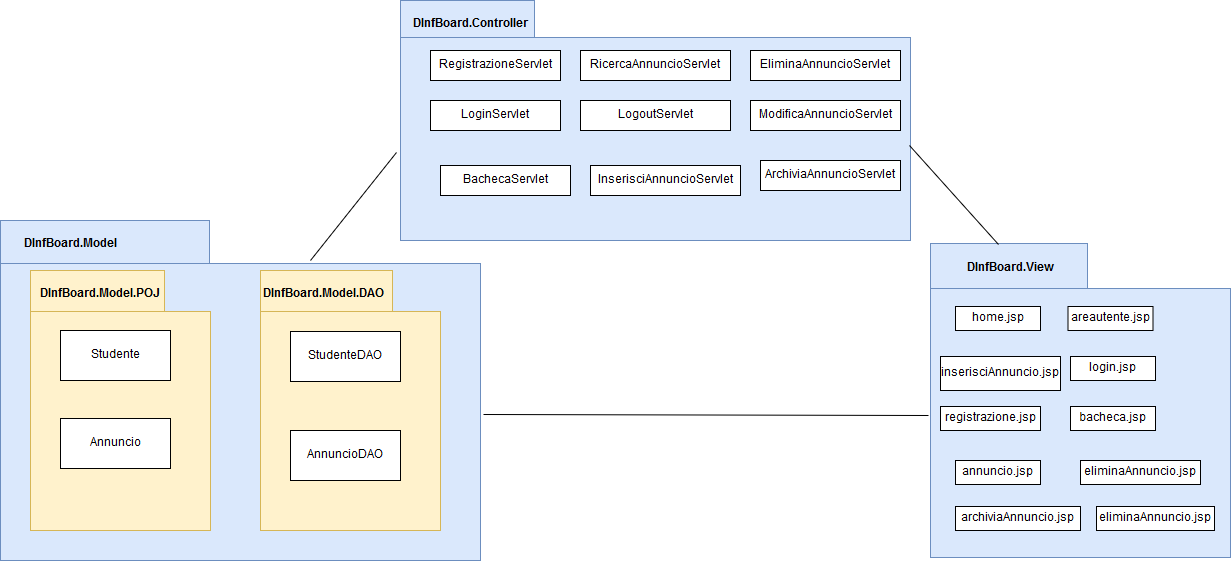
PATTERN ADAPTER



**Antonio Junior Citarella**

# Packages

**Diagramma dei Package**



**Antonio Junior Citarella**

I package che utilizzeremo nell’organizzazione del sistema sono organizzati in modo da essere riconducibili immediatamente all’elemento logico dell’architettura che essi implementano.

* Il package **DInfBoard.controller** conterrà tutte le Servlet Java che verranno richiamate per effettuare le operazioni richieste dall’utente e che si occuperanno di restituire a quest’ultimo i dati corretti.
* Il package **DInfBoard.model** conterrà le classi Java che ci permetteranno di gestire i dati persistenti del database. All’interno di esso troviamo una ulteriore divisione:

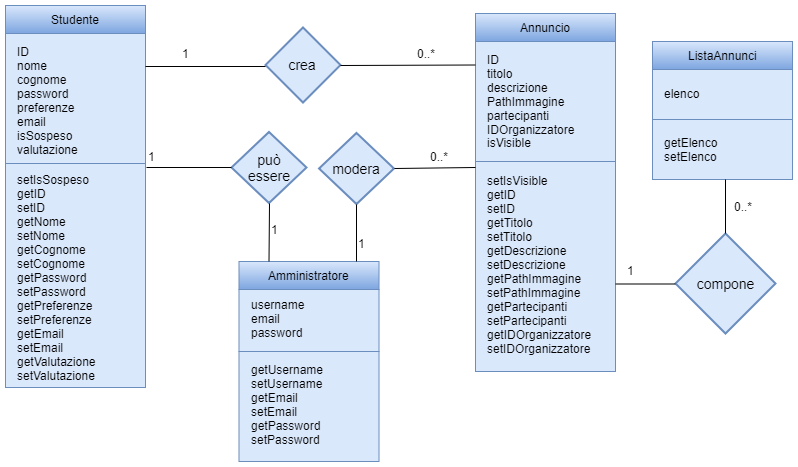
Il package **DInfBoard.model.POJ** che conterrà solo classi Plain Old Java e che ci permetterà di gestire in maniera semplice i dati prelevati dal database.Il package **DInfBoard.model.DAO** conterrà invece tutti gli oggetti DAO (Database Access Object) che si occuperanno di effettuare l’operazione di prelevamento dei dati persistenti dal DBMS e inserire questi ultimi all’interno dell’oggetto Java corrispondente.

* Il package **DInfBoard.view** conterrà invece tutti file JSP che verranno presentati all’utente e che consisteranno dell’interfaccia grafica della piattaforma.

|  |  |
| --- | --- |
| Nome | Descrizione |
| controller.user.LoginServlet | Classe Serlvet che permette di effettuare il login sulla piattaforma |
| controller.user.RegistrazioneServlet | Classe Servlet che permette di registrarsi sulla piattaforma |
| controller.user.LogoutServlet | Classe Servlet che permette ad utente di effettuare il logout dalla piattaforma |
| controller.user.ModificaProfiloServlet | Classe Servlet per modificare le credenziali di un utente sulla piattaforma |
| controller. user.RecuperaPasswordServlet | Classe Servlet per recuperare la password di un utente in caso quest’ultimo la dimentichi |
| controller.user.ChangePasswordServlet | Classe Servlet per modificare la password. |
| controller.user.ValutazioneServlet | Classe Servlet per la valutazione di uno studente. |
| controller.user.EmailSenderServlet | Classe Servlet per l’invio di una mail in caso di attività associate. |
| controller.annuncio.InserisciAnnuncioServlet | Classe Servlet per inserire un annuncio sulla piattaforma |
| controller. annuncio.ModificaAnnuncioServlet | Classe Servlet che permette di modificare un annuncio |
| controller.annuncio.EliminaAnnuncioServlet | Classe Servlet che permette di rimuovere un annuncio permanentemente dal database della piattaforma, chiedendo la valutazione dell’attività |
| controller.annuncio.ArchiviaAnnuncioServlet | Classe Servlet che permette di rimuovere un annuncio dalla bacheca della piattaforma |
| controller.annuncio.EliminaPartecipazioneServlet | Servlet che permette di rimuovere la partecipazione di un utente da un annuncio |
| controller.annuncio.BachecaServlet | Classe Servlet che ci permette di ottenere tutti gli annunci presenti sulla bacheca della piattaforma |
| controller.annuncio.CancellaAnnuncioServlet | Classe Servlet che elimina un annuncio permanentemente dal database |
| controller.annuncio.PartecipazioneServlet | Classe Servlet per la specifica della partecipazione ad un annuncio |
| controller.annuncio.RicercaAnnuncioServlet | Permette di ricercare un annuncio tra quelli disponibili sulla bacheca |
| controller.admin.InserisciAmministratoreServlet | Permette di inserire un amministratore sulla piattaforma |
| controller.admin.EliminaAmministratoreServlet | Classe Servlet che permette di eliminare un amministratore dalla piattaforma |
| controller.admin.GestionePreferenzeServlet | Permette di gestire le preferenze presenti |
| controller.admin.EliminaAnnuncioAdminServlet | Classe Servlet che permette di rimuovere un annuncio da parte di un amministratore |
| controller.admin.SospendiUtenteServlet | Classe Servlet che permette ad un amministratore di sospendere un utente dalla piattaforma |
| model.PJO.Studente | Classe per gestire gli studenti della piattaforma |
| model.PJO.Annuncio | Classe che gestisce gli annunci presenti |
| model.PJO.ListaAnnunci | Classe per gestire un elenco di annunci |
| model.PJO.PasswordUtils | Classe con metodi utili alla gestione delle password |
| model.PJO.Sender | Classe per l’invio una mail di testo |
| model.PJO.Preferenza | Classe per la gestione delle preferenze. |
| model.DAO.StudenteDao | Classe che permette di effettuare operazioni riguardanti gli studenti sul database |
| model.DAO.AnnuncioDao | Classe che permette di effettuare operazioni riguardanti gli annunci sul database |
| model.DAO.ListaAnnunciDao | Classe che permette di effettuare operazioni sul database riguardanti elenchi di annunci |
| model.DAO.AdminDao | Classe per gestire gli amministratori della piattaforma sul database |
| model.DAO.PreferenzaDao | Classe per gestire le preferenze per gli annunci sul database |
| model.DAO.GenericDao | Classe contenente perazioni comuni alla gestione del database. |
| model.DAO.implement.ManagerAdmin | Classe con i metodi per il controllo degli amministratori nel database |
| model.DAO.implement.ManagerAnnuncio | Classe con i metodi per il controllo degli annunci nel database |
| model.DAO.implement.ManagerPreferenza | Classe con i metodi per il controllo delle preferenze nel database |
| model.DAO.implement.ManagerStudente | Classe con i metodi per il controllo degli studenti nel database |

**Antonio Junior Citarella**

# Interfacce delle classi



**Antonio Vitale**

SERVLET

PACKAGE controller.admin

EliminaAmministratoreServlet.java

**import** java.io.IOException;

**import** javax.servlet.ServletException;

**import** javax.servlet.annotation.WebServlet;

**import** javax.servlet.http.HttpServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

@WebServlet("/EliminaAmministratoreServlet")

**public** **class** EliminaAmministratoreServlet **extends** HttpServlet {

**public** EliminaAmministratoreServlet() {

}

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

}

**protected** **void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

}

}

EliminaAnnuncioAdminServlet.java

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet("/EliminaAnnuncioAdminServlet")

public class EliminaAnnuncioAdminServlet extends HttpServlet {

public EliminaAnnuncioAdminServlet() {

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

}

}

GestionePreferenzeServlet.java

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet("/GestionePreferenzeServlet")

public class GestionePreferenzeServlet extends HttpServlet {

public GestionePreferenzeServlet() {

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

}

}

InserisciAmministratoreServlet.java

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet("/InserisciAmministratoreServlet")

public class InserisciAmministratoreServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

public InserisciAmministratoreServlet() {

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

}

}

SospendiUtenteServlet.java

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet("/SospendiUtenteServlet")

public class SospendiUtenteServlet extends HttpServlet {

public SospendiUtenteServlet() {

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

}

}

PACKAGE controller. annuncio

ArchiviaAnnuncioServlet.java

package controller.annuncio;

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet("/ArchiviaAnnuncioServlet")

public class ArchiviaAnnuncioServlet extends HttpServlet {

public ArchiviaAnnuncioServlet() {

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

}

}

BachecaServlet.java

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet("/BachecaServlet")

public class BachecaServlet extends HttpServlet {

public BachecaServlet() {

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

}

}

EliminaAnnuncioServlet.java

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet("/EliminaAnnuncioServlet")

public class EliminaAnnuncioServlet extends HttpServlet {

public EliminaAnnuncioServlet() {

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

}

}

EliminaPartecipazioneServlet.java

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet("/EliminaPartecipazioneServlet")

public class EliminaPartecipazioneServlet extends HttpServlet {

public EliminaPartecipazioneServlet() {

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

}

}

InserisciAnnuncioServlet.java

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet("/InserisciAnnuncioServlet")

public class InserisciAnnuncioServlet extends HttpServlet {

public InserisciAnnuncioServlet() {

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

}

}

ModificaAnnuncioServlet.java

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet("/ModificaAnnuncioServlet")

public class ModificaAnnuncioServlet extends HttpServlet {

public ModificaAnnuncioServlet() {

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

}

}

RicercaAnnuncioServlet.java

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet("/RicercaAnnuncioServlet")

public class RicercaAnnuncioServlet extends HttpServlet {

public RicercaAnnuncioServlet() {

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

}

}

PACKAGE controller.user

LoginServlet.java

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet("/RicercaAnnuncioServlet")

public class RicercaAnnuncioServlet extends HttpServlet {

public RicercaAnnuncioServlet() {

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

}

}

LogoutServlet.java

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet("/LogoutServlet")

public class LogoutServlet extends HttpServlet {

public LogoutServlet() {

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

}

}

ModificaProfiloServlet.java

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet("/ModificaProfiloServlet")

public class ModificaProfiloServlet extends HttpServlet {

public ModificaProfiloServlet() {

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

}

}

RecuperaPasswordServlet.java

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet("/RecuperaPasswordServlet")

public class RecuperaPasswordServlet extends HttpServlet {

public RecuperaPasswordServlet() {

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

}

}

RegistrazioneServlet.java

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet("/RegistrazioneServlet")

public class RegistrazioneServlet extends HttpServlet {

public RegistrazioneServlet() {

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

}

}

CLASS

PACKAGE model.PJO

public class Admin {

private String username;

private String email;

private String password;

public Admin(String username,String email,String password) {

setUsername(username);

setEmail(email);

setPassword(password);

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

}

Studente.java

**public** **class** Studente {

**public** Studente(**int** ID,String nome,String cognome,**int** preferenza,String email,String password,**boolean** isAdmin,**boolean** isSospeso,**int** valutazione) {

}

**public** **int** getID() {

}

**public** **void** setID(**int** iD) {

}

**public** String getNome() {

}

**public** **void** setNome(String nome) {

}

**public** String getCognome() {

}

**public** **void** setCognome(String cognome) {

}

**public** **int** getPreferenza() {

}

**public** **void** setPreferenza(**int** preferenza) {

}

**public** String getEmail() {

}

**public** **void** setEmail(String email) {

}

**public** String getPassword() {

}

**public** **void** setPassword(String password) {

}

**public** **boolean** isIsAdmin() {

}

**public** **void** setIsAdmin(**boolean** isAdmin) {

}

**public** **boolean** isIsSospeso() {

}

**public** **void** setIsSospeso(**boolean** isSospeso) {

}

**public** **int** getValutazione() {

}

**public** **void** setValutazione(**int** valutazione) {

}

}

Annuncio.java

**public** **class** Annuncio {

**public** Annuncio(**int** ID,String titolo,String descrizione,String PathImmagine,**int** partecipanti,**int** ID\_Organizzatore,boolean isVisible) {

}

public boolean getIsVisible() {

}

public void setVisible(boolean isVisible) {

}

**public** **int** getID() {

}

**public** **void** setID(**int** iD) {

}

**public** String getTitolo() {

}

**public** **void** setTitolo(String titolo) {

}

**public** String getDescrizione() {

**return** descrizione;

}

**public** **void** setDescrizione(String descrizione) {

**this**.descrizione = descrizione;

}

**public** String getPathImmagine() {

**return** PathImmagine;

}

**public** **void** setPathImmagine(String pathimmagine) {

PathImmagine = pathimmagine;

}

**public** **int** getPartecipanti() {

**return** partecipanti;

}

**public** **void** setPartecipanti(**int** partecipanti) {

**this**.partecipanti = partecipanti;

}

**public** **int** getIDOrganizzatore() {

**return** IDOrganizzatore;

}

**public** **void** setIDOrganizzatore(**int** iD\_Organizzatore) {

IDOrganizzatore = iD\_Organizzatore;

}

}

ListaAnnunci.java

**public** **class** ListaAnnunci {

**private** ArrayList<Annunci> elenco;

**public** ListaAnnunci() {

}

**public** ArrayList<Annunci> getElenco() {

}

**public** **void** setElenco() {

}

}

PACKAGE model.DAO

import model.PJO.\*;

public class AdminDao {

public AdminDao() {

}

public boolean inserisciAdmin(Admin a) {

return true;

}

public boolean eliminaAdmin(Admin a) {

return true;

}

public boolean eliminaAnnuncio(Annuncio a) {

return true;

}

public boolean sospendiUtente(Studente s) {

return true;

}

}

StudenteDao.java

**import** model.PJO.Studente;

**public** **class** StudenteDao {

**public** StudenteDao() {

}

**public** **boolean** Registrazione(Studente s) {

**return** **true**;

}

**public** **boolean** Login(Studente s) {

**return** **true**;

}

**public** **boolean** ModificaProfilo(Studente s) {

**return** **true**;

}

**public** **boolean** EliminaProfilo(Studente s) {

**return** **true**;

}

**public** **boolean** Valutazione(Studente s) {

**return** **true**;

}

}

AnnuncioDao.java

**import** model.PJO.\*;

**public** **class** AnnuncioDao {

}

**public** **boolean** inserisciAnnuncio(Annuncio a) {

}

**public** **boolean** rimuoviAnnuncioPersistente(Annuncio a) {

}

**public** **boolean** annuncioNonVisibile(Annuncio a) {

}

**public** **boolean** modificaAnnuncio(Annuncio a) {

}

**public** **boolean** eliminaPartecipazione(Studente s,Annuncio a) {

}

ListaAnnunciDao.java

**public** **class** ListaAnnunciDao {

**public** ListaAnnunciDAO() {

}

**public** ListaAnnunci getBacheca() {

}

**public** ListaAnnunci getAllAnnunci() {

}

**public** Annuncio getAnnuncio(ListaAnnunci l) {

}

**public** ListaAnnunci getAllAnnunciStudente(Studente s) {

}

}

Sender.java

import java.util.Date;

import java.util.Properties;

import javax.mail.Authenticator;

import javax.mail.Message;

import javax.mail.MessagingException;

import javax.mail.PasswordAuthentication;

import javax.mail.Session;

import javax.mail.Transport;

import javax.mail.internet.AddressException;

import javax.mail.internet.InternetAddress;

import javax.mail.internet.MimeMessage;

public class Sender {

public void sendPlainTextEmail(String toAddress , String subject , String message) throws AddressException, MessagingException {

}

}

**Antonio Junior Citarella**

|  |  |
| --- | --- |
| Nome classe | LoginServlet |
| Descrizione | La classe è una servlet che riceve la richiesta di accesso da parte di un utente e provvede a dargli i permessi per effettuare operazioni sulla piattaforma. |
| Pre-condizione | **context** LoginServlet:: get() **pre:** username!=null && password!=null  **context** LoginServlet:: post() **pre:** username!=null && password!=null |
| Post-condizione | **context** LoginServlet::get() **post:** session.getAttribute(“auth”)!=null && session.getAttribute(“auth”)==true |
| Invarianti |  |

**Antonio Junior Citarella**

|  |  |
| --- | --- |
| Nome classe | RegistrazioneServlet |
| Descrizione | La classe è una servlet che riceve la richiesta di effettuare l’operazione di registrazione da parte di un utente ospite. |
| Pre-condizione | **context** RegistrazioneServlet:: get() **pre:** username!=null && password!=null  **context** RegistrazioneServlet:: post() **pre:** username!=null && password!=null email!=null && preferenze!=null && nome!=null && cognome!=null |
| Post-condizione | **context** RegistrazioneServlet::get() **post:** session.getAttribute(“auth”)!=null && session.getAttribute(“auth”)==true  **context** RegistrazioneServlet:: post() **post:** session.getAttribute(“auth”)!=null && session.getAttribute(“auth”)==true |
| Invarianti |  |

**Antonio Junior Citarella**

|  |  |
| --- | --- |
| Nome classe | LogoutServelt |
| Descrizione | La classe è una servlet che riceve la richiesta di effettuare l’operazione di logout da parte di un utente precedentemente autenticato |
| Pre-condizione | **context** LogoutServlet:get() **pre**: username!=null && password!=null  **context** LogoutServlet:post() **pre:** username!=null && password!=null |
| Post-condizione |  |
| Invarianti |  |

Francesco Di Lascio

|  |  |
| --- | --- |
| Nome classe | ModificaProfiloServlet |
| Descrizione | La classe è una servlet che riceve la richiesta di effettuare operazioni di modifica sui dati precedentemente registrati da parte dell’utente |
| Pre-condizione | **context** ModificaProfiloServlet:get() **pre**: username!=null && password!=null  **context** ModificaProfiloServlet:post() **pre:** nome!=null && cognome!=null && username!=null && password!=null && preferenze!=null |
| Post-condizione |  |
| Invarianti |  |

Francesco Di Lascio

|  |  |
| --- | --- |
| Nome classe | RecuperaPasswordServlet |
| Descrizione | La classe è una servlet che riceve la richiesta di effettuare l’operazione di recupero password da parte di un utente registrato |
| Pre-condizione | **context** RecuperaPasswordServlet:get() **pre**: username=! null && email!=null  **context** RecuperaPasswordServlet:post() **pre:** username=! null && email!=null |
| Post-condizione | password != @pre password |
| Invarianti |  |

Francesco Di Lascio

|  |  |
| --- | --- |
| Nome classe | ArchiviaAnnuncioServlet |
| Descrizione | La classe è una servlet che riceve la richiesta di effettuare l’operazione di archiviazione di un annuncio da parte di un utente registrato |
| Pre-condizione | **context**ArchiviaAnnuncioServlet:get() **pre**: id!=null  **context** ArchiviaAnnuncioServlet:post() **pre:** id!= null |
| Post-condizione | **context** ArchiviaAnnuncioServlet:get() **post:** !annuncio.isVisible()  **context** ArchiviaAnnuncioServlet:post() **post** !annuncio.isVisible() |
| Invarianti |  |

Francesco Di Lascio

|  |  |
| --- | --- |
| Nome classe | InserisciAnnuncioServlet |
| Descrizione | La classe è una servlet che riceve la richiesta di effettuare l’operazione di inserimento di un annuncio da parte di un utente registrato |
| Pre-condizione | **context** InserireAnnuncioServlet:get() **pre**: titolo!=null && descrizione!= null && pathimmagine!= null  **context** InserireAnnuncioServlet:post() **pre:** titolo!=null && descrizione!= null && pathimmagine!= null |
| Post-condizione |  |
| Invarianti |  |

Francesco Di Lascio

|  |  |
| --- | --- |
| Nome classe | ModificaAnnuncioServlet |
| Descrizione | La classe è una servlet che riceve la richiesta di effettuare l’operazione di moficare un annuncio precedentemente inserito da parte di un utente registrato |
| Pre-condizione | **context** ModifcaAnnuncioServlet:get() **pre**: titolo!=null && descrizione!= null && pathimmagine!= null  **context** ModificaAnnuncioServlet:post() **pre:** titolo!=null && descrizione!= null && pathimmagine!= null |
| Post-condizione |  |
| Invarianti |  |

Francesco Di Lascio

|  |  |
| --- | --- |
| Nome classe | EliminaAnnuncioServelt |
| Descrizione | La classe è una servlet che riceve la richiesta di effettuare l’operazione di eliminazione di un annucio dalla piattaforma da parte dell’utente |
| Pre-condizione | **context** EliminaAnnuncioServlet:get() **pre**: titolo!=null && descrizione!= null && pathimmagine!= null  **context** EliminaAnnuncioServlet:post() **pre:** titolo!=null && descrizione!= null && pathimmagine!= null |
| Post-condizione |  |
| Invarianti |  |

Francesco Di Lascio

|  |  |
| --- | --- |
| Nome classe | EliminaPartecipazioneServlet |
| Descrizione | La classe è una servlet che riceve la richiesta di effettuare l’eliminazione dalla lista dei partecipanti di un annuncio precedentemente inserita |
| Pre-condizione | **context** EliminaPartecipazioneServlet:get() **pre**: titolo!=null && descrizione!= null && pathimmagine!= null && partecipanti!=null  **context** EliminaPartecipazioneServlet:post() **pre:** titolo!=null && descrizione!= null && pathimmagine!= null && partecipanti!=null |
| Post-condizione | **context** EliminaPartecipazioneServlet:get() **post** partecipanti=@pre partecipanti--;  **context** EliminaPartecipazioneServlet:post()**post** partecipanti=@pre partecipanti--; |
| Invarianti |  |

Francesco Di Lascio

|  |  |
| --- | --- |
| Nome classe | BachecaServelt |
| Descrizione | La classe è una servlet che riceve la richiesta di visualizzazione di tutti gli annunci presenti sulla bacheca |
| Pre-condizione | **context** BachecaServlet:get() **pre**: elenco!=null  **context** BachecaServlet:post() **pre:** elenco!=null |
| Post-condizione |  |
| Invarianti |  |

Francesco Di Lascio

|  |  |
| --- | --- |
| Nome classe | RicercaAnnuncioServlet |
| Descrizione | La classe è una servlet che riceve la richiesta di effettuare l’operazione di ricerca di un annuncio sulla piattaforma da parte di un utente registrato |
| Pre-condizione | **context** RicercaAnnuncioServlet:get() **pre**: titolo!=null  **context** RicercaAnnuncioServlet:post() **pre:** titolo!=null && elenco!=null |
| Post-condizione |  |
| Invarianti |  |

Francesco Di Lascio

|  |  |
| --- | --- |
| Nome classe | InserisciAmministratoreServlet |
| Descrizione | La classe è una servlet che riceve la richiesta di inserimento di un nuovo amministratore da parte di un amministratore della piattaforma |
| Pre-condizione | **context** InserisciAmministratoreServlet:get() **pre**: username!=null  **context** InserisciAmministratoreServlet:post() **pre:** username!=null && email!= null |
| Post-condizione |  |
| Invarianti |  |

Francesco Di Lascio

|  |  |
| --- | --- |
| Nome classe | EliminaAmministratoreServlet |
| Descrizione | La classe è una servlet che riceve la richiesta di eliminare un amministratore da parte di un amministratore della piattaforma |
| Pre-condizione | **context** EliminaAmministratoreServlet:get() **pre**: username!=null  **context** EliminaAmministratoreServlet:post() **pre:** username!=null && email!=null |
| Post-condizione |  |
| Invarianti |  |

Francesco Di Lascio

|  |  |
| --- | --- |
| Nome classe | GestionePreferenzeServlet |
| Descrizione | La classe è una servlet che riceve la richiesta di gestire le preferenze presenti sulla piattaforma |
| Pre-condizione | **context** GestionePreferenzeServlet:get() **pre**: nome!=null && descrizione!=null  **context** GestionePreferenzeServlet:post() **pre:** nome!=null && descrizione!=null |
| Post-condizione |  |
| Invarianti |  |

Francesco Di Lascio

|  |  |
| --- | --- |
| Nome classe | EliminaAnnuncioAdminServlet |
| Descrizione | La classe è una servlet che riceve la richiesta di effettuare l’eliminazione di un annuncio da parte di un amministratore della piattaforma |
| Pre-condizione | **context** EliminaAnnuncioAdminServlet:get() **pre**: elenco!=null  **context** EliminaAnnuncioAdminServlet:post() **pre:** |
| Post-condizione |  |
| Invarianti |  |

Francesco Di Lascio

|  |  |
| --- | --- |
| Nome classe | SospendiUtenteServlet |
| Descrizione | La classe è una servlet che riceve la richiesta di effettuare l’operazione di sospensione di un utente registrato da parte di un amministratore della piattaforma |
| Pre-condizione | **context** SospendiUtenteServlet:get() **pre**: username!=null && email!=null  **context** SospendiUtenteServlet:post() **pre:** username!=null && email!= null |
| Post-condizione | **context** SospendiUtenteServlet:get() **post** utente.isSospeso==true  **context** SospendiUtenteServlet:post() **post** utente.isSospeso==true |
| Invarianti |  |

Francesco Di Lascio

|  |  |
| --- | --- |
| Nome classe | AdminDao |
| Descrizione | La classe è un Database Access Object che ci fornirà l’accesso al DBMS per effettuare le operazioni che riguardano un amministratore |
| Pre-condizione | **context** AdminDao::inserisciAdmin(Admin a) **pre: a**!=null && a.getUsername!=null && a.getPassword!=null && a.getEmail!=null  **context**AdminDao::eliminaAdmin (Admin a) **pre:**  a!=null && a.getUsername && a.getPassword!=null && a.getEmail!=null  **context** AdminDao::eliminaAnnuncio (Annuncio a) **pre: a**!=null  **context** AdminDao::sospendiUtente (Studente s) **pre:** s!=null s.getID()!=null |
| Post-condizione |  |
| Invarianti |  |

Antonio Junior Citarella

|  |  |
| --- | --- |
| Nome classe | StudenteDao |
| Descrizione | La classe è un Database Access Object che ci fornirà l’accesso al DBMS per effettuare le operazioni che riguardano un utente |
| Pre-condizione | **context** StudenteDao::login(Studente s) **pre:** s!=null  **context** StudenteDao::registrazione (Studente s) **pre:**  s!=null && s.username && s.password!=null && s.email!=null && s.nome!=null && s.cognome!=null  **context** StudenteDao::modificaProfilo(Studente s) **pre:** s!=null && s.username && s.password!=null && s.email!=null && s.nome!=null && s.cognome!=null  **context** StudenteDao::eliminaProfilo(Studente s) **pre:** s!=null  **context** StudenteDao::valutazione(Studente s,int valutazione) **pre:** s!=null && valutazione>=0 |
| Post-condizione | **Context** StudenteDao::valutazione(Studentes,int valutazione,int numeroAnnunciOrganizzati) **post:** [s.valutazione=@pre s.valutazione](mailto:s.valutazione=@pre%20s.valutazione) + valutazione / numeroAnnunciOrganizzati) |
| Invarianti |  |

**Antonio Junior Citarella**

|  |  |
| --- | --- |
| Nome classe | AnnuncioDao |
| Descrizione | La classe è un Database Access Object che ci fornirà l’accesso al DBMS per effettuare le operazioni che riguardano un annuncio |
| Pre-condizione | **context** AnnuncioDao::inserisciAnnuncio(annuncio a) **pre:** a!=null && a.getID()!=null && a.getTitolo()!=null && a.getDescrizione!=null  **context** AnnuncioDao::modificaAnnuncio(annuncio a) **pre:** a!=null && a.getID()!=null && a.getTitolo()!=null && a.getDescrizione!=null  **context** AnnuncioDao::eliminaPartecipazione (Studente s,annuncio a) **pre:** a!=null && a.getID()!=null && a.getTitolo()!=null && a.getDescrizione!=null && s!=null && s.getID!=null |
| Post-condizione | **context** AnnuncioDao::eliminaPartecipazione (Studente,annuncio a) **post:** [a.partecipanti=@pre](mailto:a.partecipanti=@pre) a.partecipanti --; |
| Invarianti |  |

**Antonio Junior Citarella**

|  |  |
| --- | --- |
| Nome classe | ListaAnnunciDao |
| Descrizione | La classe è un Database Access Object che ci fornirà l’accesso al DBMS per effettuare le operazioni che riguardano un annuncio |
| Pre-condizione | **context** ListaAnnunciDao::getAnnuncio (ListaAnnucni l) **pre:** l!=null  **context** ListaAnnunciDao::getAllAnnunciStudente(Studente s) **pre:** s!=null |
| Post-condizione |  |
| Invarianti |  |

**Antonio Junior Citarella**

|  |  |
| --- | --- |
| Nome classe | Admin |
| Descrizione | La classe è un Plain Java Object che ci fornisce una mappatura di una tupla della tabella ‘Admin’ all’interno del database |
| Pre-condizione | **context**Admin::Admin (int ID,String username,String password,String email) **pre:**username!=null && password!=null && email!=null |
| Post-condizione |  |
| Invarianti |  |

**Antonio Junior Citarella**

|  |  |
| --- | --- |
| Nome classe | Annuncio |
| Descrizione | La classe è un Plain Java Object che ci fornisce una mappatura di una tupla della tabella ‘Annuncio’ all’interno del database |
| Pre-condizione | **context** Annuncio::Annuncio(int ID,String titolo,String descrizione,String PathImmagine,int partecipanti,int ID\_Organizzatore) **pre:** descrizione!=null && PathImmagine!=null && partecipanti>=0 |
| Post-condizione |  |
| Invarianti |  |

**Antonio Junior Citarella**

|  |  |
| --- | --- |
| Nome classe | Studente |
| Descrizione | La classe è un Plain Java Object che ci fornisce una mappatura di una tupla della tabella ‘Studente’ all’interno del database |
| Pre-condizione | **context** Studente::Studente (int ID,String username,String password,String nome,String cognome,String preferenza, String email,boolean isAdmin,boolean isSospseso) **pre:** username!=null && password!=null && nome!=null && cognome!=null &&email!=null |
| Post-condizione |  |
| Invarianti |  |

**Antonio Junior Citarella**

|  |  |
| --- | --- |
| Nome classe | ListaAnnunci |
| Descrizione | La classe gestisce oggetti java che rappresentano l’elenco degli annunci che verranno visualizzati all’interno della bacheca |
| Pre-condizione | **context** ListaAnnunci::getElenco () **pre:** elenco!=null |
| Post-condizione | **context** ListaAnnunci::ListaAnnunci() elenco.size()==0 |
| Invarianti |  |

**Antonio Junior Citarella**

|  |  |
| --- | --- |
| Nome classe | Sender |
| Descrizione | La classe gestisce oggetti java che rappresentano l’elenco |
| Pre-condizione | **context** ListaAnnunci::sendTextEmail (String toAddress, String subject, String message) **pre:** toAddress!=null && subject!=null && message!=null && !(toAddress.equals(“”)) && !(subject.equals(“”)) && !(message.equals(“”)) |
| Post-condizione |  |
| Invarianti |  |

**Antonio Junior Citarella**