Immagine che contiene Carattere, testo, logo, cerchio

Il contenuto generato dall'IA potrebbe non essere corretto.

PROGETTAZIONE DI UN CLASS DIAGRAM VOLTO ALL’IMPLEMENTAZIONE DI UN HACKATHON <https://github.com/galetizia/ProgettoObject.git>

*Autore:*

*Corso di OBJECT ORIENTATION*

*Gabriele Letizia N86005267*

*Fabio Iannicelli N86005383*

*ANNO ACCADEMICO 2024/2025*

[***PROGETTAZIONE DEL CLASS DIAGRAM*** 2](#_Toc196390814)

[**1.1 | Introduzione** 2](#_Toc196390815)

[**1.2 | Class Diagram** 3](#_Toc196390816)

[***DOCUMENTAZIONE RELATIVA ALLE CLASSI*** 3](#_Toc196390817)

[**2.1 | Classe Hackathon** 3](#_Toc196390818)

[**2.2 | Classe Team** 4](#_Toc196390819)

[**2.3 | Classe Aggiornamento** 5](#_Toc196390820)

[**2.4 | Classe Utente** 5](#_Toc196390821)

[**2.5 | Classe Giudice** 6](#_Toc196390822)

[**2.6 | Classe Organizzatore** 6](#_Toc196390823)

[**2.7 | Classe Voto (Classe associata)** 7](#_Toc196390824)

# ***PROGETTAZIONE DEL CLASS DIAGRAM***

## **1.1 | Introduzione**

Un hackathon, ovvero una "maratona di hacking", è un evento durante il quale, vari team di partecipanti si sfidano per progettare e implementare nuove soluzioni basate su una certa tecnologia o mirate a un certo ambito applicativo.

## **1.2 | Class Diagram**

Immagine che contiene testo, schermata, diagramma, linea

Il contenuto generato dall'IA potrebbe non essere corretto.

# ***DOCUMENTAZIONE RELATIVA ALLE CLASSI***

## **2.1 | Classe Hackathon**

Nell’implementazione della classe “Hackathon” , sono stati inseriti vari attributi per descrivere ogni singola edizione.

Ogni Hackathon ha un titolo identificativo con relativa sede, ha una data di inizio e di fine, con una data massima di iscrizione (solitamente due giorni prima della data di inizio).

Inoltre, avremo un massimo numero di iscritti, e una massima dimensione per ogni team, sono state poi inserite delle liste dove verranno salvati tutti i partecipanti, tutti i team e tutti i giudici.

Metodi della classe:

* **MostraTeams()** : stampa a video la lista dei team presenti in quell’edizione dell’Hackathon.
* **AggiungiTeam(team : TEAM)** : passiamo a questo metodo un team che verrà aggiunto alla lista dei team presenti all’Hackathon.
* **IscriviUtente(utente : UTENTE)** : passiamo a questo metodo un utente che verrà aggiunto alla lista degli utenti iscritti all’Hackathon.
* **Int GetMaxDimTeam()** : restituisce la dimensione massima per i team.
* **Int GetMaxIscritti()** : restituisce il numero massimo di iscrizioni possibili all’Hackathon.
* **Boolean PrenotazioniAperte()** : restituisce TRUE se non è stato superato il termine massimo di iscrizione all’Hackathon, FALSE altrimenti.
* **String GetProblema()** : restituisce la traccia del problema, assegnato dai giudici, che i team dovranno risolvere.
* **PubblicaClassifica(voti : List<VOTO>)** : stampa a video i team con la relativa media di voti.
* **StampaGiudici()** : stampa a video la lista relativa ai giudici assegnati all’Hackathon.

La classe “Hackathon” è in relazione con la classe “Organizzatore”, “Utente”, “Giudice”, “Team”.

* Un hackathon può avere soltanto un organizzatore.
* Un hackathon può avere N utenti.
* Un hackathon può avere N giudici.
* Un hackathon può avere N team.

## **2.2 | Classe Team**

Nella classe “Team” sono presenti gli attributi utili per identificare un gruppo di lavoro.

Ogni Team ha un nome specifico, una dimensione massima specifica passata dall’hackathon, una lista dei componenti, la media dei voti ricevuti e la lista dei voti ricevuti da ogni giudice.

Metodi della classe:

* **AggiungiMembro(utente : UTENTE)** : aggiunge un utente alla lista dei membri del team.
* **String GetNome()** : restituisce il nome del team al chiamante.
* **StampaTeam()** : stampa a video il nome di uno specifico team e i relativi membri.
* **Boolean isPieno()** : restituisce TRUE se il team ha raggiunto il numero massimo di componenti, FALSE altrimenti.
* **Boolean isVuoto()** : restituisce TRUE se il team non ha componenti, FALSE altrimenti.
* **AGGIORNAMENTO creaAggiornamento(nome : String, documento : String)** : crea un nuovo aggiornamento implementato dal team**.**
* **EspelliMembro(utente : UTENTE)** : elimina uno specifico membro dalla lista dei componenti del team.
* **StampaVoti()** : stampa a video la lista dei voti ricevuti dai giudici.

La classe “Team” è in relazione con le classi “Hackathon”, “Aggiornamento”, “Utente ”, “Giudici”.

* Un team può partecipare ad una sola hackathon.
* Un team può creare più aggiornamenti.
* Un team può avere più utenti.
* Un team può essere valutato da più giudici.

## **2.3 | Classe Aggiornamento**

Nella classe “Aggiornamento” sono presenti gli attributi per identificare uno specifico aggiornamento.

Ogni Aggiornamento ha un nome specifico, il documento caricato dal team (file) contenente l’aggiornamento sviluppato e ogni aggiornamento può essere commentato da un giudice.

La classe non presenta metodi al di fuori del costruttore.

La classe “Aggiornamento” è in relazione con le classi “Giudice” e “Team”.

* Un aggiornamento può essere commentato da N giudici.
* Un aggiornamento può essere associato ad un solo team.

## **2.4 | Classe Utente**

Nella classe “Utente” andiamo a specificare gli attributi utili all’identificazione di uno specifico utente.

Ogni utente avrà un nome ed un cognome, l’email personale, il ruolo (appena iscritto alla piattaforma sarà automaticamente contrassegnato come partecipante), avrà poi uno username ed una password.

Metodi della classe:

* **Boolean login(username : String, password : String)** : Restituisce TRUE se il login ha avuto successo, FALSE altrimenti.
* **String GetNome()** : Restituisce il nome dell’utente.
* **String GetCognome()** : Restituisce il cognome dell’utente.

La classe “Utente” è in relazione con le classi “Hackathon”, “Team”.

* Un utente può partecipare ad un solo Hackathon.
* Un utente può far parte di un solo team.

La classe “Utente” ha due classi figlie, la classe “Giudice” e la classe “Organizzatore”.

## **2.5 | Classe Giudice**

Nella classe “Giudice” non avremo attributi aggiuntivi oltre a quelli ereditati della classe padre “Utente”.

Metodi della classe:

* **ValutaTeam(team : TEAM, punteggio : int)** : Attribuisce un voto aggiuntivo alla lista dei voti presente nella classe “Team”.
* **String CommentaAggiornamento(a : AGGIORNAMENTO, commento : String)** : Attribuisce un commento ad un aggiornamento caricato da un team.
* **PubblicaProblema(problema : String, hackathon : HACKATHON)** : Carica nella classe “Hackathon” il problema che i team dovranno risolvere.

La classe “Giudice” è in relazione con le classi “Aggiornamento”, “Team” ed “Hackathon”.

* Un giudice può commentare più aggiornamenti.
* Un giudice può valutare più team.
* Un giudice può partecipare ad un solo Hackathon.

## **2.6 | Classe Organizzatore**

Nella classe “Organizzatore” non avremo attributi aggiuntivi oltre a quelli ereditati della classe padre “Utente”.

Metodi della classe:

* **GIUDICE AggiungiGiudice(utente : UTENTE, partecipanti : List<UTENTE>, hackathon : HACKATHON)** : Aggiorna il ruolo di uno specifico utente promuovendolo a giudice.

La classe “Organizzatore” è in relazione con la classe “Hackathon”

* Un organizzatore può gestire un solo Hackathon.

## **2.7 | Classe Voto (Classe associata)**

La classe “Voto” è una classe associata alla relazione di valutazione tra la classe “Giudice” e la classe “Team”.

Nella classe “Voto” sono presenti gli attributi utili alla valutazione di un team; ogni voto può avere una valutazione da 1 a 10, lo specifico giudice che ha assegnato la relativa valutazione ed il relativo team a cui è stato attribuito il voto.

Metodi della classe:

* **Int GetValutazione()** : Restituisce un valore intero corrispondente alla valutazione.
* **GIUDICE GetGiudice()** : Restituisce il giudice che ha assegnato il voto.
* **TEAM GetTeam()** : Restituisce il team a cui è stato assegnato il voto.

La classe “Voto” non ha relazioni poiché si tratta di una classe associata.