**Documentazione Class Diagram**

Per quanto riguarda il class diagram su Visual Paradigm, dopo aver letto ed esaminato la traccia abbiamo trovato le seguenti classi. Hackathon, ovvero l’evento che si svolge dopo un certo periodo di tempo, Utente ovvero colui che partecipa all’evento suddiviso tramite una generalizzazione in Organizzatore, colui che organizza l’evento, Giudice colui che giudica ed assegna voti ai Team dei partecipanti, raggruppati in team e infine Partecipante, colui che partecipa all’evento. Abbiamo individuato come classe anche Team ovvero il gruppo di partecipanti che aderiscono all’Hackathon ed infine alcune classi intermedie, cioè si trovano tra un’associazione di due classi, queste classi sono Voto, situato tra Giudice e Team, Documento situato tra Giudice e Team e alla fine Invito situato tra Giudice e Organizzatore, e all’associazione riflessiva di Partecipante. Per quanto riguarda gli attributi nella classe Utente abbiamo nome, cognome, data nascita, email e password, questi attributi verranno utilizzati per i metodi di Utente e possiedono visibilità protected in modo che possono essere accessibili ed utilizzati dalle sottoclassi. Hackathon possiede come attributi titolo, sede, data inizio, data fine, massimo partecipanti, dimensione team e organizzatore di tipo Organizzatore ovvero colui che organizza l’evento, tutti gli attributi hanno visibilità private perché non tutti possono accedere a questi attributi. Per Voto abbiamo come attributo voto cioè quello che il giudice assegna al team di visibilità protected in moda da essere utilizzate solo dalle classi nello stesso pacchetto. In Documento come attributi abbiamo data e documento dove data è la data di pubblicazione del documento e documento la relazione del team che verrà controllata e modificata dal Giudice anche qui per lo stesso motivo di voto gli attributi hanno visibilità protected. In Invito come attributi abbiamo data invito che corrisponde alla data in cui viene mandato l’invito e messaggio ovvero il messaggio d’invito, anche qui gli attributi sono di visibilità protected in modo da permettere solo alle classi all’interno dello stesso pacchetto di poter utilizzare questi attributi appartenenti alla classe Invito. Nella classe Team abbiamo trovato gli attributi nome team che corrisponde al nome del team e partecipanti di tipo List<Partecipante> ovvero le varie informazioni sui partecipanti del Team anche qui gli attributi sono protected, all’interno dello stesso pacchetto è possibile ad accedere agli attributi di questa classe. In alcune classi abbiamo implementato dei metodi, queste classi sono Utente con metodi di Registrazione perché un Utente si deve registrare per poter partecipare all’evento, questo metodo prende come parametri nome, cognome, data nascita, email e password dell’Utente e restituisce un Utente. Il Sign In dove l’Utente accede se già registrato utilizza come parametri email e password, anche qui il metodo restituisce un Utente. I metodi di chiusura e apertura iscrizione in Organizzatore perché si occupa di aprire e chiudere le iscrizioni all’evento prendono come parametro una data vedono se quella data è valida per apertura o chiusura dell’evento e restituiscono una data. Metodo descrizione problema in Giudice dove un Giudice descrive il problema da risolvere questo metodo non utilizza parametri e restituisce una stringa. Metodi media voto e classifica in Voto dove nel primo metodo si prendono tutti i voti dei giudici assegnati ad un team e si calcola la media dei voti non prende parametri e restituisce un intero, il secondo metodo classifica viene pubblicata una classifica con la media dei voti assegnati ad ogni team in ordine decrescente non prende parametri e restituisce una stringa. In Documento abbiamo il metodo modifica documento dove un Team pubblica un documento i giudici vedono il documento e dopo lo modificano prende come parametro una stringa e restituisce un documento. In Hackathon, come metodi abbiamo tutti get e set, per accesso e gestione degli attributi privati degli attributi privati. In invito come metodi abbiamo accetta e rifiuta invito siccome a Giudice e Partecipante gli arriva un invito loro possono accettare o rifiutare quell’invito, tutte e due le funzioni prendono come parametro il messaggio e restituiscono un void. Tutti i metodi hanno visibilità public così e possibile accederci anche fuori dalla classe dove vengono dichiarati. Per l’unione tra classi abbiamo individuato le seguenti associazioni, tra Organizzatore ed Hackathon perché un Organizzatore con cardinalità (1 : 1….\*) perché l’Organizzatore può organizzare da uno a più Hackathon mentre un Hackathon può essere organizzato da un solo Organizzatore. Associazione riflessiva di Partecipante, con cardinalità (\* : \*) perché un Partecipante può invitare diversi partecipanti e un Partecipante può essere invitato da più partecipanti. Associazione tra Giudice e Team con cardinalità (1…\* : 1…\*) perché un Giudice giudica da uno a più Team e questi vengono giudicati da uno a più giudici. Associazione tra Organizzatore e Giudice, anche se queste classi sono tutte e due delle generalizzazioni della classe Utente abbiamo preferito fare quest’associazione siccome viene specificato che l’Organizzatore invita i giudici a partecipare all’Hackathon e poiché abbiamo creato la classe Invito per l’associazione riflessiva di Partecipante abbiamo preferito utilizzarla anche qui come classe intermedia al posto di utilizzare un metodo di invito, quest’associazione possiede una cardinalità (\* : 1..\*) perché un Giudice vieni in vitato da più organizzatori ed un Organizzatore invita da uno a più giudici. Associazione tra Team e Hackathon con cardinalità (\* : 1….\*), perché il Team partecipa ad uno o più Hackathon e all’Hackathon partecipano più team. Infine abbiamo una composizione tra Team e Partecipante perché senza Partecipante il Team non esiste con cardinalità (1:N) perché il Partecipante partecipa ad un Team e in un Team partecipano più partecipanti.

NOTA : nel file visual paradigm la classe invito siccome non volevamo sovrapporre le linee delle associazioni e per mancanza di spazio, l’abbiamo scritta due volte ma sono la stessa classe.