UNIVERSITÀ
DI TRENTO

Dipartimento di ingegneria e scienza dell'informazione

Progetto:

Vocable

Titolo del documento:

Documento di Architettura

Document info

Doc. Name	D3-Vocable_Architettuta	Doc. Number	D3
Description	Il documento include diagrammi delle classi e codice in OCL		

- Scopo del documento

Il presente documento definisce l'architettura del progetto "Vocable" mediante diagramma delle classi in UML (Unified Modeling Language) e codice in OCL (Object Constraint Language). Nel documento precedente (D2-Vocable_SpecificaRequisiti) sono forniti: diagramma dei casi d'uso, diagramma di contesto e diagramma dei componenti. Basandosi sui suddetti diagrammi si definisce l'architettura del sistema, di cui si illustrano le classi da implementare a livello di codice e la logica che regola il funzionamento del software.

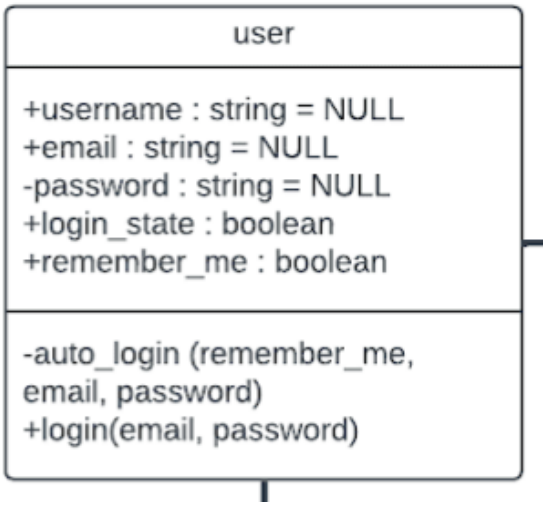
1. Diagramma delle classi

Nel presente capitolo vengono presentati le classi previste nell'ambito del progetto "Vocable". Ogni componente presente nel diagramma dei componenti diventa una o più classi. Tutte le classi individuate sono caratterizzate da un nome, una lista di attributi che identificano i dati che vengono gestiti dalla classe stessa e una lista di metodi che definiscono le operazioni interne alla classe. Ogni classe può essere associata alle altre classi e tramite questa associazione viene definita la relazione tra di esse.

Riportiamo di seguito le classi individuate a partire dai diagrammi di contesto e dei componenti. In questo processo si è proceduto anche nel massimizzare la coesione e minimizzare l'accoppiamento tra classi.

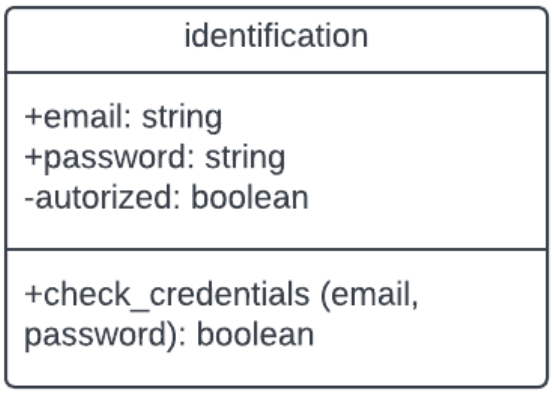
1.1 Utenti

Gli utenti dell'applicazione sono identificati dalla classe user, in essa sono contenuti i dati relativi ad ogni account di gioco (il valore di default di tali attributi è NULL, a meno che un utente registrato non selezioni l'opzione "ricordami") e le funzioni relative al login. La differenziazione tra utenti identificati e anonimi, i quali si distinguono tra loro per diverse interfacce e funzionalità, avviene attraverso l'attributo login_state e le classi astratte anon_interface e id_interface.



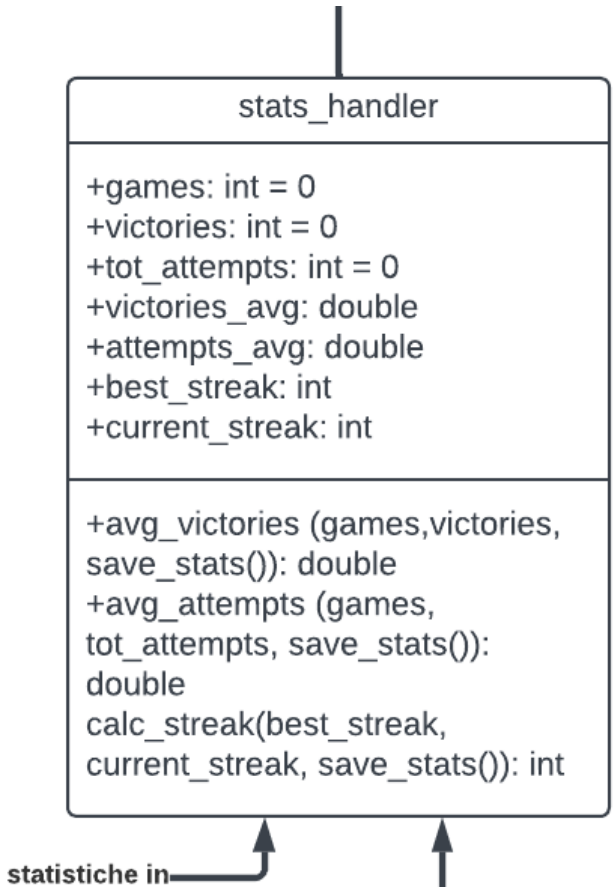
1.2 Gestione autenticazione

Il primo sistema subordinato di Vocable è quello di gestione credenziali. La classe “identification” rappresenta il confronto delle credenziali immesse dall’utente durante il login con quelle registrate nel database interno contenente le specifiche degli account di gioco Vocable.



1.3 Gestione statistiche

Vocable presenta poi un sistema interno di gestione statistiche, grazie al quale partite, vittorie, numero medio di tentativi a partita e serie di partite vincenti sono calcolate, registrate e rese disponibili per la visualizzazione ai rispettivi utenti.

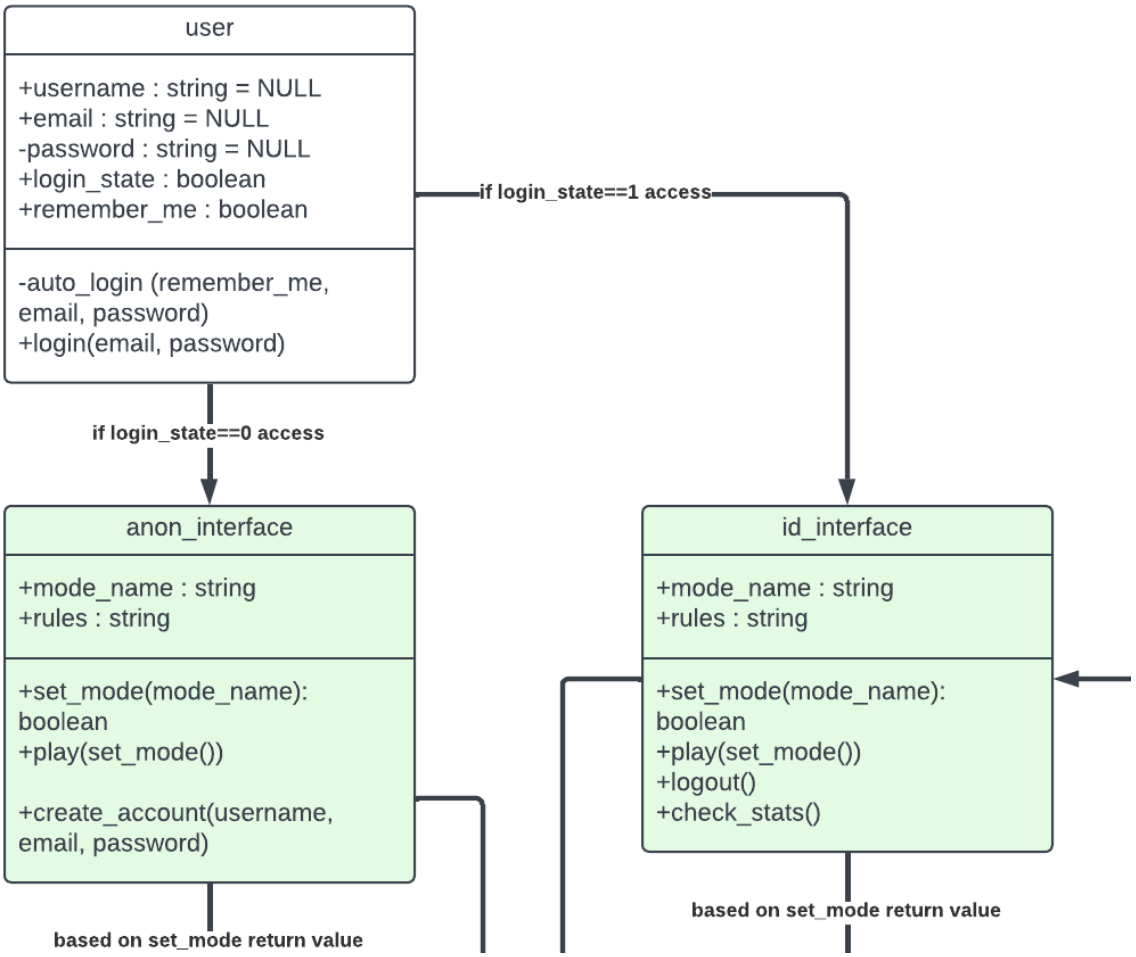


1.4 Vocabolario

Al centro della meccanica di gioco di Vocable vi è il dizionario. Un database contenente parole indicizzate, le rispettive definizioni e funzionalità (unuse_index, update_indexes) che garantiscono la non-ripetitività richiesta dal gioco.

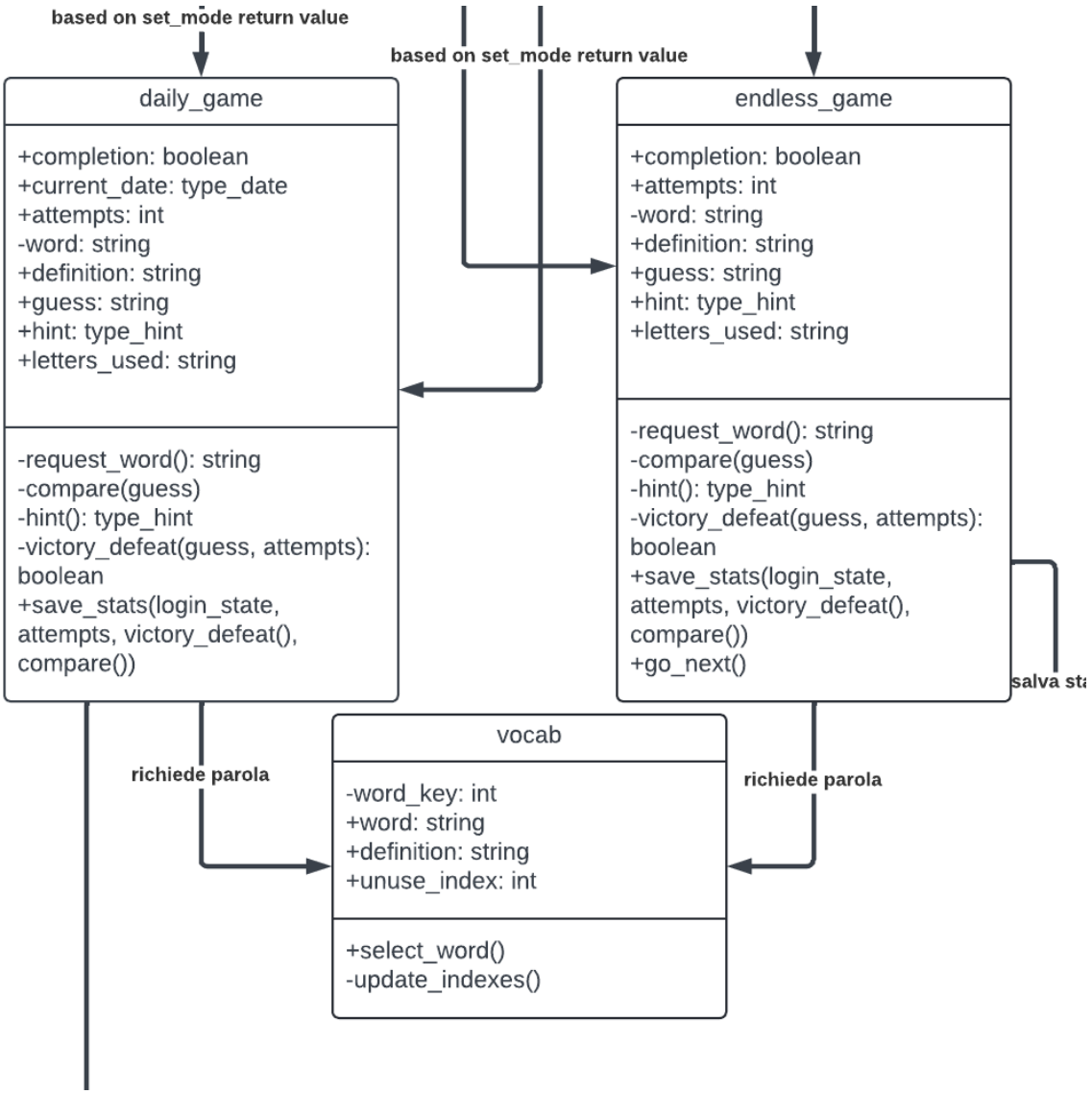
1.5 Schermate Home

Le interfacce per gli utenti anonimi o registrati sono identificate rispettivamente dalle classi astratte anon_interface e id_interface. Queste forniscono le funzioni necessarie a: selezionare una modalità di gioco, iniziare una partita, creare un account (per gli utenti anonimi) e effettuare il logout (per gli utenti identificati).



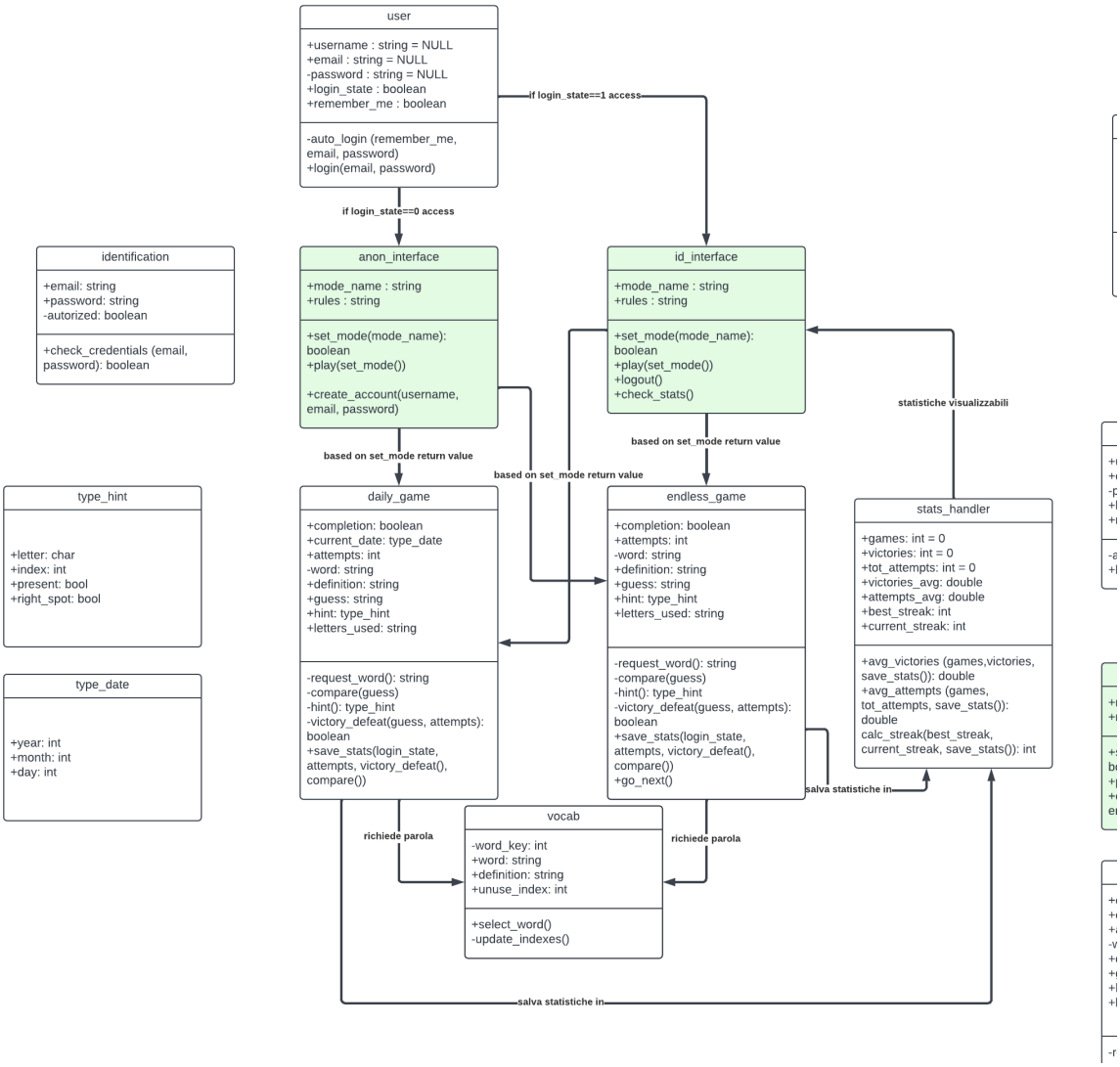
1.6 Gestione partite

Sono infine presentate le classi contenenti i metodi di gestione partita per la modalità giornaliera e illimitata. Esse forniscono agli utenti l'interfaccia di gioco e tutte le funzioni necessarie a completare una partita e a registrarne le statistiche se si dispone di un account Vocable. Tali funzioni comprendono: l'effettuazione di tentativi per indovinare la parola nascosta, la fornitura di indizi ai giocatori a ogni tentativo incorretto, l'output di notifiche di vittoria o sconfitta, il salvataggio automatico delle statistiche sull'account dell'utente registrato e la possibilità di iniziare una nuova partita in modalità illimitata.



1.7 Diagramma delle classi complessivo

In seguito è riportato il diagramma completo, costituito da tutte le classi precedentemente presentate e da alcune ausiliarie come i tipi date e hint.

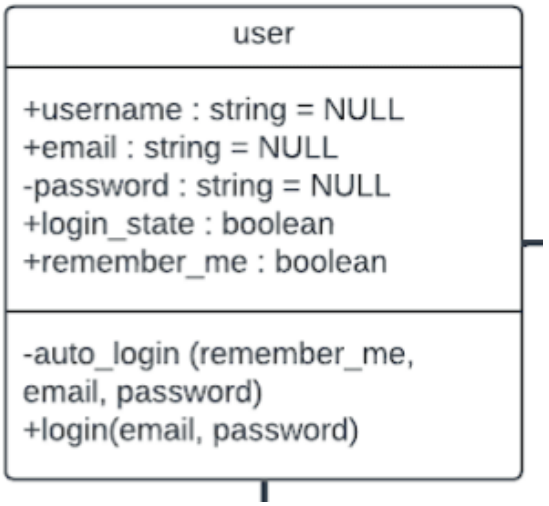


2. Codice in Object Constraint Language

In questa sezione del documento viene descritta in modo formale la logica prevista nell'ambito di alcune operazioni di alcune classi. Questa logica viene descritta tramite il linguaggio "Object Constraint Language" (OCL), poiché tali concetti non sono esprimibili in nessun altro modo formale nel contesto di UML.

2.1 Login

Nella classe User è presente il metodo login(). L'esecuzione di questo metodo comporta, oltre al "salvataggio" delle informazioni dell'utente, il cambio di stato del valore dell'attributo "login_state" a "true".
Questa condizione, presente nella classe raffigurata qui di seguito:

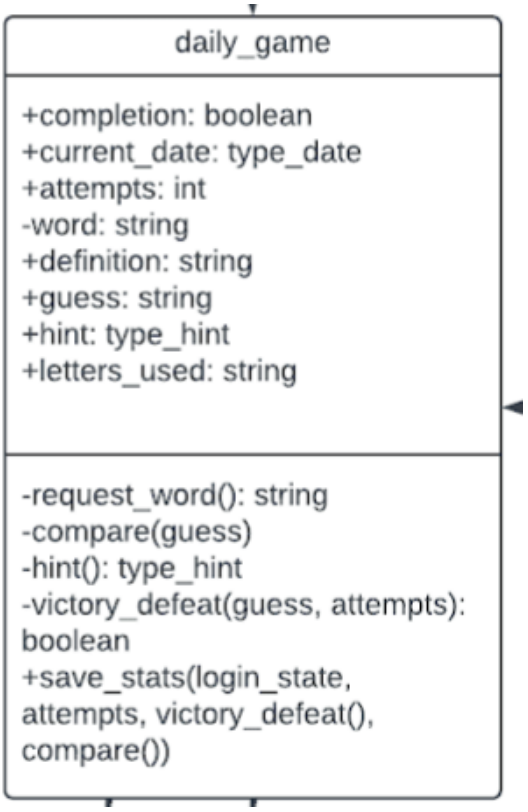


è espressa in OCL tramite una postcondizione, con questo codice:

```
context user::login(email, password)
post login_state = true
```

2.2 Condizioni per i Daily Games

Sono presenti due condizioni che devono essere sempre verificate per ogni “daily_game”: il numero dei tentativi non deve superare 6 e la lunghezza delle parole inserite dagli utenti devono essere uguali alla lunghezza della parola da indovinare. Queste condizioni fanno riferimento alla classe qui in seguito:



e sono espresse in OCL attraverso un'invariante con questo codice:

```
context daily_game inv:
attempts <= 6
strlen(guess) == strlen(word)
```

2.3 Completion in daily_games

Nella classe appena riportata (2.2), è presente anche l'attributo "completion".
Il valore di questo attributo può cambiare solo se i tentativi effettuati superano quelli disponibili o nel caso in cui l'utente indovini la parola.
Queste costrizioni sono espresse tramite un'invariate in questo modo:

```
context daily_game inv:
if(self.attempts >= 7 or victory_defeat()) then
self.completion = 1
```

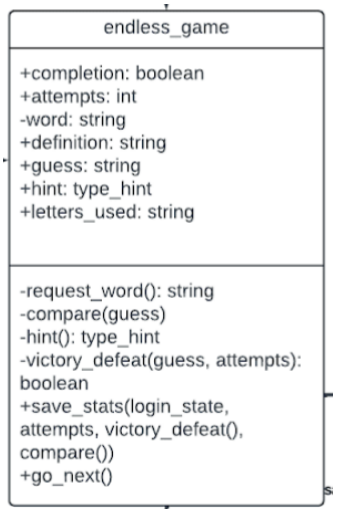
2.4 Salvataggio statistiche in daily_games

Nella stessa classe (2.2) è presente anche la funzione "save_stats". Questa può essere eseguita solo se la variabili "completion" e "login_state" sono equivalenti ad '1'.
Queste condizioni sono espresse, in OCL, in tale modo:

```
context daily_game::save_stats(login_state, attempts, victory_defeat(), compare())
pre: (self.completion = 1) AND (login_state=1)
```

2.5 Condizioni per gli Endless Games

Anche nella classe endless_game ci sono due condizioni che devono essere verificate per ogni istanza della classe. Queste sono: il numero di tentativi per indovinare la parola deve essere al massimo 6 e la lunghezza della parola da indovinare deve essere lunga quanto le parole immesse dall'utente.
La classe a cui fanno riferimento queste condizioni è la seguente:



e sono dunque esplicitate in codice OCL come il codice qui inserito:

```
context endless_game inv:
attempts <= 6
strlen(guess) == strlen(word)
```

2.6 Completion in endless_game

Nella classe appena vista (2.4), sono presenti delle costrizioni dell'attributo “completion”.
Tali costrizioni sono le seguenti: completion può assumere il valore ‘1’ solo, e solo se, l’utente indovina la parola o i tentativi per indovinare la parola in questione finiscono.
Queste costrizioni possono essere espresse in OCL in questo modo:

```
context endless_game inv:
if(self.attempts >= 7 or victory_defeat()) then
self.completion = 1
```

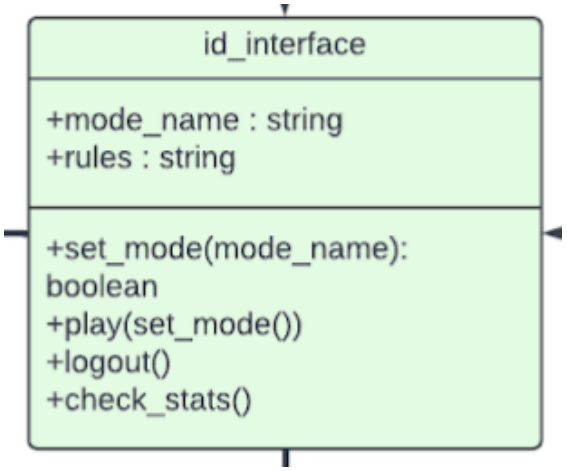
2.7 Salvataggio statistiche in endless_games

Nella stessa classe (2.5) è presente anche la funzione “save_stats”. Tale funzione può essere eseguita solo se la variabile “completion”, presente sempre nella stessa classe, corrisponde al valore ‘1’ e, in contemporanea, anche quando la variabile “login_state” è a ‘1’
Questa condizione è espressa tramite una precostrizione come la seguente:

```
context endless_game::save_stats(login_state, attempts, victory_defeat(), compare())
pre: (self.completion = 1) AND (login_state=1)
```

2.8 Logout

Nella classe "id_interface" è presente il metodo "logout()".
Questo metodo, dopo essere stato eseguito, prevede il cambiamento di stato della variabile "login_state" da vero, a falso.
Tale condizione, presente nella seguente classe:



è codificata in OCL tramite una postcondizione, scritta in questo modo:

```
context id_interface::logout()
post: login_state = false
```

3. Diagramma delle classi con codice OCL

Riportiamo, infine, il diagramma delle classi col codice OCL visto fino ad ora: