UNIVERSITY OF TRENTO

Progetto EpiOpera

Gruppo T18

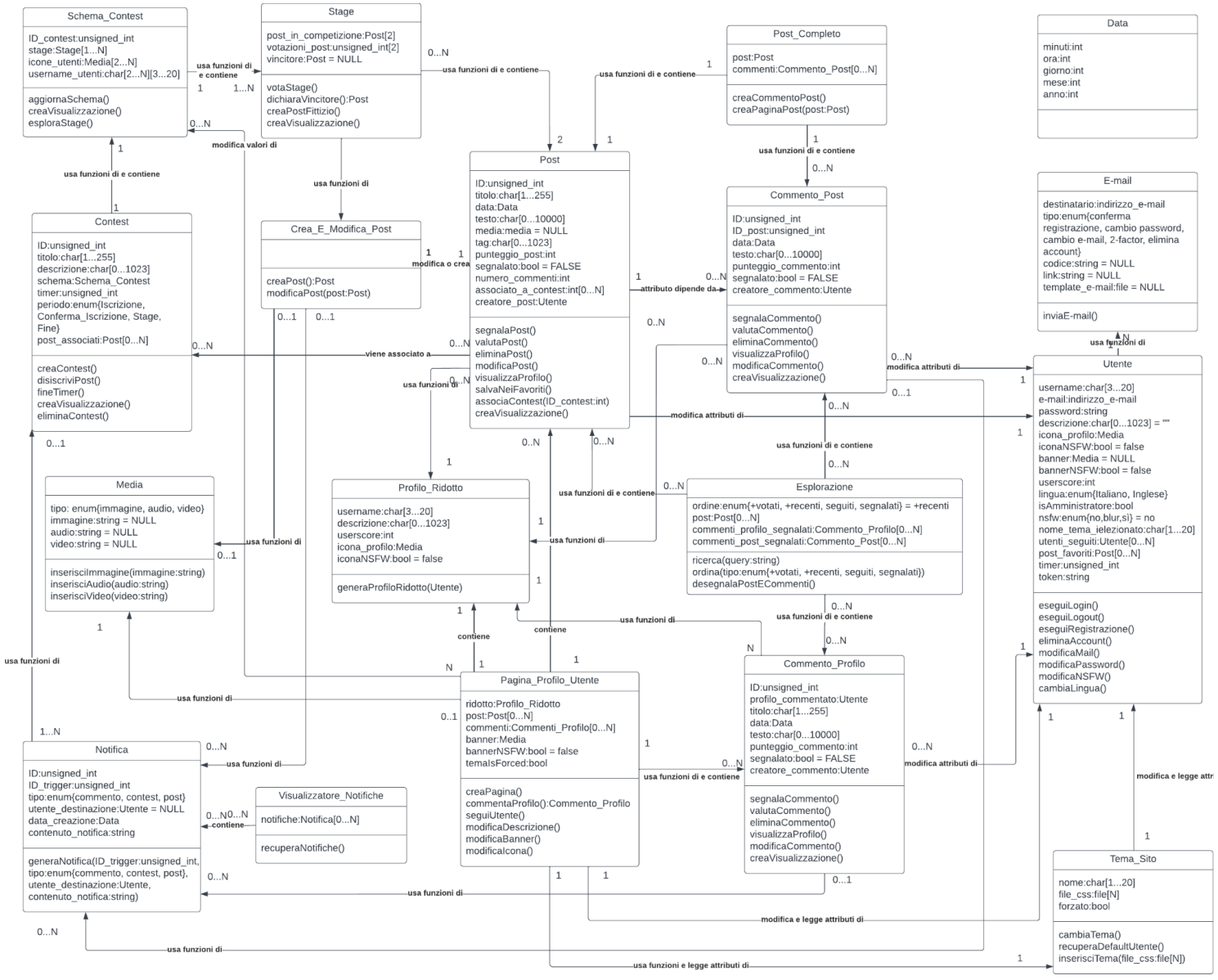
Alessandro Marostica
Nicolò Marchini
Nicolò Masellis

Indice

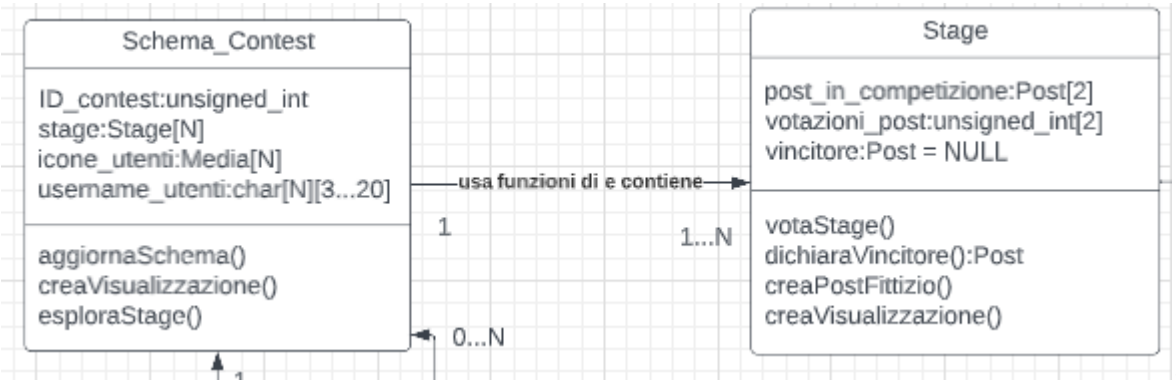
Scopo del Documento

Il presente documento riporta la definizione dell'architettura del progetto EpiOpera usando diagrammi delle classi in Unified Modeling Language (UML) e codice in Object Constraint Language (OCL). Nel precedente documento è stato presentato il diagramma degli use case, il diagramma di contesto e quello dei componenti. Ora, tenendo conto di questa progettazione, viene definita l'architettura del sistema dettagliando da un lato le classi che dovranno essere implementate a livello di codice e dall'altro la logica che regola il comportamento del software. Le classi vengono rappresentate tramite un diagramma delle classi in linguaggio UML. La logica viene descritta in OCL perché tali concetti non sono esprimibili in nessun altro modo formale nel contesto di UML.

Diagramma delle Classi



Nel presente capitolo vengono presentate le classi previste nell'ambito del progetto EpiOpera. Ogni componente presente nel diagramma dei componenti è rappresentato da una o più classi. Tutte le classi individuate sono caratterizzate da un nome, una lista di attributi che identificano i dati gestiti dalla classe e una lista di metodi che definiscono le operazioni previste all'interno della classe. Ogni classe può essere anche associata ad altre classi e, tramite questa associazione, è possibile fornire informazioni su come le classi si relazionano tra loro. In una associazione la freccia indica chi è che "riceve" l'associazione. Ad esempio questa associazione:



si deve leggere come “Uno Schema_Contest usa funzioni di e contiene uno o più Stage”.
Riportiamo di seguito le classi individuate a partire dai diagrammi di contesto e dei componenti.

Utente

Il sito userà questa classe per indicare le caratteristiche di un utente all’interno del sito.
Login contiene una variabile token che indica NULL se nessun login è stato eseguito, altrimenti contiene un token univoco all’utente che ha eseguito il login.
La classe contiene anche un timer che è inizializzato a 900 secondi quando viene eseguita la funzione eseguiLogin() o quando l’utente loggato esegue una operazione all’interno del sito, quando il timer raggiunge lo zero viene eseguita automaticamente la funzione eseguiLogout().
La visibilità globale di questa classe ci permette di sfruttarla per poter accedere a determinate informazioni dalle quali dipende la visualizzazione di altre pagine.
La funzione eseguiLogin() è descritta nel dettaglio dallo use case [Login] del D2.
La funzione eseguiLogout() è descritta nel dettaglio dallo use case [Logout] del D2.
La funzione eseguiRegistrazione() è descritta nel dettaglio dallo use case [Registrazione] del D2.
La funzione eliminaAccount() è descritta nel dettaglio dallo use case [Elimina Account] del D2.
La funzione modificaMail() è descritta nel dettaglio dallo use case [Cambio E-mail] del D2.
La funzione modificaPassword() è descritta nel dettaglio dallo use case [Cambio Password] del D2.
La funzione modificaNSFW() modifica il valore dell’attributo nsfw, questa funzione è presente per poter soddisfare il requisito non funzionale [Controllo visibilità post NSFW].
La funzione cambiaLingua() è descritta nel dettaglio dallo use case [Cambia Lingua del Sito]. del D2.
Questa classe interagisce con la classe [E-mail] per poter inviare e-mail.

E-mail

Una funzione fondamentale del sito è quella dello spedire E-mail informative riportanti informazioni riguardanti attività sul profilo.
Tale mail ha lo scopo di fornire un appoggio nei processi di login e registrazioni e saranno quindi composte di informazioni riguardo le attività del profilo associato al determinato indirizzo email.

In base al tipo di informazione che deve condividere avrà un template differente e genererà un link con una destinazione differente ed eventualmente un codice (specifico per 2FA).
Si interfaccerà poi con il Servizio E-mail per inviare le e-mail e con il Database MongoDB per ricevere il template.
Il funzionamento della funzione inviaE-mail() è descritto nel dettaglio dallo use case [Invio E-mail] del D2.

Notifica

Il sito può inviare a un utente delle notifiche informative riguardanti menzioni e informazioni sui contest in corso.
Ogni notifica avrà un insieme di informazioni specifiche all'evento che lo ha generato e l'utente sarà in grado, all'interazione con essa, di reindirizzare l'utente alla pagina in cui può interagire con l'oggetto della notifica usando un link.
Un'altra classe può richiedere la generazione di una notifica usando la funzione generaNotifica(), se il tipo della notifica da generare è contest allora la notifica non ha un singolo destinatario ma viene mandata a tutti gli utenti del sito.
Il funzionamento della funzione generaNotifica() è descritto nel componente [Genera Notifica] del D2.

Visualizzatore_Notifiche

Questa classe permette a un utente di visualizzare la lista di tutte le notifiche da lui ricevute in ordine dalla più recente.
Questo avviene interagendo con il Database MongoDB usando la sessione di login attuale dell'utente con la funzione recuperaNotifiche(), per maggiori dettagli riguardo la funzione vedi lo use case [Visualizza Notifiche] del D2.
Questa classe contiene istanze della classe [Notifica].

Profilo_Ridotto

E' una classe specifica realizzata per la funzione del sito di visualizzare un profilo in maniera "ridotta", cioè omettendo alcune informazioni che creerebbero troppa confusione in altre pagine ma comunque accessibili con un click verso la pagina completa del profilo.
A tale proposito la classe è composta dalle informazioni base di un profilo più un link che porta alla pagina del profilo stesso.
Altre classi possono richiedere la visualizzazione del profilo usando la funzione generaProfiloRidotto(), per maggiori informazioni su questa funzione vedi l'use case [Visualizza Profilo Ridotto di un Utente] nel D2.

Pagina_Profilo_Utente

Nel sito esistono pagine specifiche alla visualizzazione completa di un profilo utente.
Questa classe sfrutta l'esistenza della soprastante classe [Profilo_Ridotto] per evitare di avere funzioni e attributi ridondanti, inoltre contiene una lista di tutti i commenti associati al profilo con la classe [Commento_Profilo] e una lista di tutti i post creati dal profilo usando la classe [Post].

Contiene inoltre numerose funzioni che permettono le interazioni e le modifiche del profilo.
La funzione commentaProfilo() permette a un utente autenticato di creare una istanza della classe [Commento_Profilo] associato al profilo, per maggiori dettagli sulla funzione vedi lo use case [Commenta Profilo] del D2.
La funzione seguiUtente() permette a un utente autenticato di seguire (o smettere di seguire se lo stava seguendo già) il profilo visualizzato, per maggiori dettagli sulla funzione vedi lo use case [Segui Utente] del D2.
La funzione creaPagina() viene usata dal sito per creare e far visualizzare a un utente la pagina associata ad un profilo, in particolare interagisce con la classe [Tema_Sito] se il parametro tema IsForced è true per recuperare il tema personalizzato dell'utente a cui appartiene la pagina profilo, per maggiori dettagli sulla funzione vedi lo use case [Visualizza Profilo Completo di un Utente] del D2.
La funzione modificaDescrizione() permette all'utente possessore del profilo di modificare la descrizione del suo profilo, per maggiori dettagli sulla funzione vedi lo use case [Modifica Profilo] del D2.
La funzione modificaBanner() permette all'utente possessore del profilo di modificare il banner del suo profilo interagendo con la classe [Media] e aggiornando gli attributi anche all'interno della istanza della classe [Utente] relativa all'utente che detiene la pagina profilo. Un utente amministratore può usare questa funzione anche su profili altrui ma in modo ristretto, per maggiori dettagli sulla funzione vedi lo use case [Modifica Profilo] del D2.
La funzione modificaIcona() permette all'utente possessore del profilo di modificare l'icona del suo profilo interagendo con la classe [Media], andando anche a modificare l'istanza associata a tale icona all'interno di eventuali istanze di classe [Contest] e aggiornando gli attributi anche all'interno della istanza della classe [Utente] relativa all'utente che detiene la pagina profilo. Un utente amministratore può usare questa funzione anche su profili altrui ma in modo ristretto, per maggiori dettagli sulla funzione vedi lo use case [Modifica Profilo] del D2.

Post

Funzione centrale del sito è l'esplorazione di contenuti caricati da altri utenti, è quindi necessaria una classe Post che contenga le informazioni sul contenuto.
In particolare la classe Post si riferisce alla visualizzazione ridotta dei post che viene utilizzata da tutte quelle parti del sito che richiedono la visualizzazione di un post.
La visualizzazione di un post cambia a seconda della presenza o meno del tag #NSFW al suo interno e delle impostazioni dell'utente che ha eseguito il login (se non ci sono sessioni di login allora di default i post NSFW vengono nascosti).
L'attributo numero_commenti viene ricavato interrogando il Database MongoDB riguardo il numero di commenti associato a un post.
La funzione segnalaPost() viene eseguita da un utente autenticato e segnala un post per la revisione mettendo a true l'attributo segnalato, per maggiori dettagli sulla funzione vedi lo use case [Segnala Post] del D2.
La funzione valutaPost() viene eseguita da un utente autenticato che assegna un voto a un post modificando l'attributo punteggio_post, per maggiori dettagli sulla funzione vedi lo use case [Valuta Post] del D2.
La funzione eliminaPost() viene eseguita dall'utente che ha creato il post in questione o da un utente amministratore, elimina il post dal Database MongoDB, per maggiori dettagli sulla funzione vedi lo use case [Elimina Post] del D2.

La funzione visualizzaProfilo() esegue la funzione generaProfiloRidotto() della classe [Profilo_Ridotto].
La funzione salvaNeiFavoriti() viene eseguita da un utente autenticato che aggiunge questo post alla sua lista post_favoriti, per maggiori dettagli sulla funzione vedi lo use case [Salva Post nei Preferiti] del D2.
La funzione associaContest() viene eseguita dall'utente creatore del post che associa il post ad una istanza della classe [Contest], per maggiori dettagli sulla funzione vedi lo use case [Commenta Post] del D2.
La funzione creaVisualizzazione() viene eseguita da una parte del sito quando vuole visualizzare un post, per maggiori dettagli sulla funzione vedi lo use case [Visualizza Post Ridotto] del D2.
La funzione modificaPost() viene eseguita dall'utente creatore del post o da un utente amministratore e richiama la funzione modificaPost() della classe [Crea_E_Modifica_Post].

Post_Completo

Questa classe serve ad unire in un unico luogo i commenti associati a un post e il post stesso.
La funzione creaCommentoPost() viene eseguita da un utente autenticato che crea un'istanza della classe [Commento_Post] e la comunica al Database MongoDB, per maggiori dettagli sulla funzione vedi lo use case [Commenta Post] del D2.
La funzione creaPaginaPost() viene usata dal sito per creare la visualizzazione di un post completo, questa funzione richiama la funzione creaVisualizzazione() della istanza della classe [Post] che si vuole vedere e creaVisualizzazione() delle istanze della classe [Commento_Post] associate al post, per maggiori dettagli sulla funzione vedi lo use case [Visualizza Post Completo] del D2.

Esplorazione

Questa classe si rende necessaria vista la funzione di esplorazione che fornisce una pagina di scorrimento dei post ordinati in diverse maniere. Nel caso specifico di un utente amministratore questa pagina ha la seconda valenza di pagina utile allo scorrimento dei post segnalati e offre quindi funzioni utili alla moderazione. Inoltre permette la ricerca dei contenuti in essa mostrati.

Crea_E_Modifica_Post

All'utente che intende caricare contenuto è fornita una pagina che faciliti il processo, è quindi necessaria una classe con le funzioni utili specificamente ad inviare il post al database o a modificarlo.
Esiste inoltre una classe Media creata al momento del caricamento del contenuto. Questa viene spesso utilizzata in altre pagine per fare riferimento al contenuto in maniera unificata.

Gestione Contest

Caratteristica distintiva del sito EpiOpera è la possibilità di partecipare a contest sia in quanto contribuente che in ruolo di giudicante. A tale scopo viene creata una classe Contest

contenente le informazioni principali del contest. Un attributo specifico di questa classe è un'ulteriore classe nominata Schema_Contest, cioè la rappresentazione grafica in sé del contest, dotata di funzioni dinamiche in grado di generare una visualizzazione accurata a schermo. Ulteriormente questa seconda classe sfrutta la classe Stage, cioè il contesto specifico di due post in competizione, è il componente che si occupa di registrare votazioni, tenere traccia del vincitore del duello e creare post fittizi per questioni di ordine.

Contest

La classe Contest è specifica per l'omonima funzionalità che contraddistingue il sito. Contiene chiavi di identificazione del contest, informazioni primarie e referenze ai contenuti inclusi. Le funzioni che ha a disposizione sono principalmente legate alla macro-gestione di eventi principali che possono eventualmente verificarsi nel decorso del contest. Contiene inoltre un attributo di tipo Schema_Contest, descritto sotto.

Schema_Contest

La classe contiene tutte le informazioni necessarie alla corretta visualizzazione dell'intera struttura del contest, inclusi contenuti e relativi autori. Le funzioni a disposizione di tale classe riguardano la visualizzazione e la costruzione di tale schema. Tale schema è inoltre costruito basandosi sulla classe Stage, sotto descritta.

Stage

La classe stage gestisce l'entità minima della competizione all'interno del contest. Descrive il "duello" in corso tra due post e tiene traccia delle valutazioni, a questo fine sfrutta la classe Post. Le funzioni che ha a disposizione sono legate alla votazione, alla visualizzazione e alla logistica del contest.

Gestione Commenti

Parte delle interazioni è la possibilità di commentare sia post che profili degli utenti. A questo scopo sono create le classi Commento_Profilo e Commento_Post, autoesplicative, differiscono solo per un attributo riguardante il loro contenuto. Esse sono dotate di attributi contenenti le informazioni, il contenuto e delle chiavi che le legano al contenuto di destinazione. Contengono inoltre funzioni riguardanti la modifica, eliminazione e moderazione.

Commento_Profilo

La classe contiene, oltre al contenuto in sé, numerosi attributi che riguardano informazioni su autore, tempo di pubblicazione, interazioni e chiavi di identificazione. Oltre alle funzioni di modifica ed eliminazione a disposizione dell'utente standard, la classe dispone di funzioni relative alla moderazione.

Commento_Post

Similmente alla classe Commento_Profilo, questa classe dispone, oltre al contenuto in sé, numerosi attributi che riguardano informazioni su autore, tempo di pubblicazione, interazioni e chiavi di identificazione. Oltre alle funzioni di modifica ed eliminazione a disposizione dell'utente standard, la classe dispone di funzioni relative alla moderazione. A differenza di Commento_Profilo manca dell'attributo titolo e invece di relazionarsi a profili si relaziona a post.

Tema_Sito

Questa classe gestisce la visualizzazione da parte di tutto il sito dei temi.
Si occupa anche di permettere ad un utente autenticato di aggiungere un tema personalizzato al sito.

Media

Questa classe gestisce le informazioni riguardante i media non testuali presenti nel sito.

Data

Informazioni principali riguardanti numerosi contenuti sono data e ora di pubblicazione o modifica, viene quindi creata una classe Data contenente tali informazioni.

OCL

Profilo Ridotto

La classe necessita che il media dell'immagine di profilo sia esclusivamente un'immagine.

context Profilo_Ridotto : self.icona_profilo.tipo = immagine

Post Completo

Il commento può essere creato esclusivamente da un utente autenticato.

context Post_Completo :: creaCommentoPost()
pre: Utente.token != NULL

Pagina di Esplorazione

Nella pagina di esplorazione un utente amministratore può visualizzare una lista dei post segnalati, questo è quindi un requisito, un utente autenticato semplice può invece visualizzare una lista di post in base all'ordine selezionato.

context Esplorazione :: ordina()

pre: if (!Utente.isAmministratore AND self.ordine = segnalati) then (self.ordine = +recenti)

context Esplorazione :: desegnalaPostECommenti()
pre: Utente.isAmministratore

Pagina Profilo Utente

Il banner del profilo utente deve essere un'immagine, per poter seguire un profilo l'utente deve essere autenticato.
context Pagina_Profilo_Utente : self.banner.tipo = immagine

context Pagina_Profilo_Utente :: seguiUtente()
pre: Utente.token != NULL

context Pagina_Profilo_Utente :: modificaDescrizione()
pre: Utente.username = self.ridotto.username

context Pagina_Profilo_Utente :: modificaBanner()
pre: Utente.username = self.ridotto.username OR Utente.isAmministratore

context Pagina_Profilo_Utente :: modificaIcona()
pre: Utente.username = self.ridotto.username OR Utente.isAmministratore

Crea E Modifica Post

Per creare un post un utente deve essere autenticato, inoltre per modificare un post il nome dell'utente deve corrispondere al nome dell'autore del post.

context Crea_E_Modifica_Post :: creaPost()
pre: Utente.token != NULL

context Crea_E_Modifica_Post :: modificaPost()
pre: Utente = self.post.creatore_post

Media

La particolare combinazione di attributi all'interno della classe permette di stabilire il tipo del media inserito.

context Media:
if (tipo = immagine) then (immagine != NULL AND audio = NULL AND video = NULL)
else if (tipo = audio) then (immagine = NULL AND audio != NULL AND video = NULL)
else if (tipo = video) then (immagine = NULL AND audio = NULL AND video != NULL)

context Media :: inserisciImmagine(immagine:string)
post: self.tipo = immagine AND self.immagine != NULL

context Media :: inserisciAudio(audio:string)
post: self.tipo = audio AND self.audio != NULL

context Media :: inserisciVideo(video:string)
post: self.tipo = video AND self.video != NULL

Contest

context Contest :: creaContest()
pre: Utente.isAmministratore
post: self.timer >= 86400 AND self.timer <= 2678400

context Contest :: disiscriviPost()
pre: Utente.isAmministratore

context Contest :: eliminaContest()
pre: Utente.isAmministratore

Schema Contest

context Schema_Contest : icone_utenti[N].tipo = immagine

Stage

context Stage :: votaStage()
pre: Utente.token != NULL

Gestione Temi

La personalizzazione del tema può essere effettuata solo da un utente autenticato.

context Tema_Sito :: inserisciTema(file_css:file[N])
pre: Utente.token != NULL

Utente

Un utente autenticato può modificare solamente la propria mail, password e NSFW. Inoltre un utente autenticato può eliminare solamente il proprio account. Tuttavia un utente amministratore può eliminare un account altrui.

context Utente : icona_profilo.tipo = immagine

context Utente : banner.tipo = immagine

context Utente :: eseguiLogin()
pre: self.utente.token = NULL

post: self.utente.token != NULL

context Utente :: eseguiLogout()
pre: self.utente.token = NULL
post: self.utente.token != NULL

context Utente :: eseguiRegistrazione()
pre: self.utente.token = NULL

context Utente :: eliminaAccount()
pre: self.token != NULL OR self.isAmministratore

context Utente :: modificaMail()
pre: self.token != NULL

context Utente :: modificaPassword()
pre:self.token != NULL

context Utente :: modificaNSFW()
pre:self.token != NULL

Gestione Commenti

La creazione, la modifica e l'interazione con i commenti possono essere effettuate solo da utenti autenticati e, ove necessario, sotto la specifica condizione di proprietà del commento. Inoltre, tuttavia, un utente amministratore ha la possibilità di rimuovere commenti al fine di moderazione.

context Commento_Profilo :: eliminaCommento()
pre: Utente = self.creatore_commento OR Utente.isAmministratore

context Commento_Profilo :: modificaCommento()
pre: Utente = self.creatore_commento

context Commento_Profilo :: segnalaCommento()
pre: Utente.token != NULL
post: self.segnalato = true

context Commento_Profilo :: votaCommento()
pre: Utente.token != NULL

context Commento_Post :: eliminaCommento()
pre: Utente = self.creatore_commento OR Utente.isAmministratore

context Commento_Post :: modificaCommento()
pre: Utente = self.creatore_commento

context Commento_Post :: segnalaCommento()

pre: Utente.token != NULL
post: self.segnalato = true

context Commento_Post :: votaCommento()
pre: Utente.token != NULL

Post Ridotto

La creazione e l'interazione con i post possono essere effettuate solo da utenti autenticati e, ove necessario, sotto la specifica condizione di proprietà del commento. Inoltre, tuttavia, un utente amministratore ha la possibilità di rimuovere post al fine di moderazione.

context Post :: eliminaPost()
pre: Utente = self.creatore_post OR Utente.isAmministratore

context Post :: segnalaPost()
pre: Utente.token != NULL
post: self.segnalato = true

context Post :: valutaPost()
pre: Utente.token != NULL

context Post :: salvaNeiFavoriti()
pre: Utente.token != NULL

context Post :: commentaPost()
pre: Utente.token != NULL