



Definizione Di Prodotto

Informazioni sul documento

Titolo documento	Definizione Di Prodotto
Versione attuale	v1.0.0
Data versione attuale	2012/03/20
Data creazione	2012/02/15
Redazione	Giacomo Lorigiola Stefano Faoro Luca Guerra
Revisione	Antonio Pretto[v1.0.0]
Approvazione	Luca Lorenzini
Stato documento	Formale
Uso	Esterno
Distribuito da	SevenFold
Destinato a	Prof. Tullio Vardanega Dott. Amir Baldissera referente Mentis s.r.l. Dott.ssa Elisa Sartore referente Mentis s.r.l.

Sommario

Questo documento contiene la struttura del sistema Woty, analizzando nel dettaglio i suoi componenti.

Diario delle modifiche

Versione	Data	Autore	Modifiche
v1.0.0	2012/03/20	Luca Lorenzini	Approvazione e rilascio prima versione
v0.14.0	2012/03/20	Luca Guerra	Aggiornamento sezione server
v0.13.0	2012/03/18	Stefano Faoro	Aggiornata e completata sezione mobile
v0.12.0	2012/03/17	Giacomo Lorigiola	Aggiornamenti sezione desktop
v0.11.1	2012/03/15	Stefano Faoro	Correzioni ortografiche
v0.11.0	2012/03/10	Luca Guerra	Aggiunta sezione in sottosistema server
v0.10.0	2012/03/07	Luca Guerra	Sistemata struttura server
v0.9.0	2012/02/25	Stefano Faoro	Correzione specifica componenti Android
v0.8.0	2012/02/17	Giacomo Lorigiola	Completata specifica componenti Desktop
v0.7.0	2012/02/16	Stefano Faoro	Inserita specifica componenti sistema Android
v0.6.0	2012/02/16	Giacomo Lorigiola	Inserita specifica componenti Manager e Client nel sistema Desktop
v0.5.0	2012/02/15	Luca Guerra	Inserita specifica classe "Resource" e relative sottoclassi
v0.4.0	2012/02/15	Giacomo Lorigiola	Inserita struttura e grafici Desktop
v0.3.0	2012/02/14	Stefano Faoro	Inserita struttura sistema Mobile
v0.2.0	2012/02/13	Luca Guerra	Prima stesura struttura documento e inserimento struttura sistema Server
v0.1.0	2012/02/13	Stefano Faoro	Creazione documento

Indice

1	Introduzione	7
1.1	Scopo del documento	7
1.2	Scopo del prodotto	7
1.3	Glossario	7
1.4	Riferimenti	7
1.4.1	Normativi	7
1.4.2	Informativi	7
2	Standard di progetto	8
2.1	Standard di progettazione architetturale	8
2.2	Standard di documentazione del codice	8
2.3	Standard di denominazione di entità e relazioni	8
2.4	Standard di programmazione	8
2.5	Strumenti di lavoro	8
3	Specifiche Woty Server	9
3.1	Introduzione	9
3.2	Wildcard	9
3.3	Proprietà del linguaggio Ruby	9
3.4	Implementazione standard di una risorsa	9
3.4.1	Model	9
3.4.2	Controller	10
3.4.3	View	10
3.4.4	Fogli di stile	11
3.4.5	Javascript	11
3.5	Comportamenti e necessità anomale	11
3.5.1	Decorator	11
3.5.2	Helper	11
3.5.3	Gerarchie standard	11
3.6	Specifiche risorse framework	11
3.6.1	Classe ApplicationController	12
3.6.2	Classe ActiveRecord::Base	12
3.7	Specifiche risorse standard	13
3.7.1	User	13
3.7.2	SuperUser	15
3.7.3	DesktopUser	15
3.7.4	MobileUser	16
3.7.5	Admin	16
3.7.6	Customer	17
3.7.7	WorkGroup	18
3.7.8	Plan	18
3.7.9	Quest	19
3.7.10	QuestDescription	20
3.7.11	TextDescription	21
3.7.12	ImageDescription	21
3.7.13	VideoDescription	22
3.7.14	QuestChallenge	22
3.7.15	ComboChallenge	22
3.7.16	TrueFalseChallenge	23
3.7.17	MultiOptionChallenge	23
3.7.18	Achievement	24
3.7.19	Like	25
3.7.20	Comment	25
3.7.21	StreamElement	26

3.7.22	Specifica Classe QuestHistory	27
3.7.23	Ticket	27
3.7.24	CompletedAchievement	28
3.7.25	CompletedAchievementElement	29
3.7.26	Condition	29
3.7.27	CustomQuest	30
3.7.28	QuestAssociation	30
3.7.29	QuestCategory	30
3.7.30	SharedQuest	31
3.7.31	SignupElement	31
3.7.32	StatusChangeElement	31
3.7.33	Wgcategorie	32
3.8	Altre risorse	32
3.8.1	XmlrpcController	32
3.8.2	Autenticazione	33
3.8.3	Autorizzazione	34
3.8.4	PagesController	35
3.8.5	Moduli	36
4	Specifiche Woty Desktop	36
4.1	Diagramma Componenti	36
4.2	Diagramma Classi	37
4.3	Specifiche componente Manager	38
4.3.1	Specifiche classe Manager	39
4.3.2	Specifiche classe Exception	42
4.4	Specifiche componente Client	43
4.4.1	Specifiche classe Client	43
4.4.2	Specifiche libreria LibMaia	44
4.5	Specifiche componente Account	45
4.5.1	Specifiche classe LoginWindow	45
4.5.2	Specifiche classe Account	46
4.5.3	Specifiche libreria simpleCrypt	48
4.5.4	Specifiche classe Preferences	48
5	Specifiche Woty Mobile	51
5.1	Diagramma Componenti	51
5.2	Diagramma delle Classi	52
5.3	Specifiche componente Model	53
5.3.1	com.woty.android.model.Resource	53
5.3.2	com.woty.android.model.User	54
5.3.3	com.woty.android.model.Challenge	56
5.3.4	com.woty.android.model.TrueFalseChallenge	57
5.3.5	com.woty.android.model.ComboChallenge	58
5.3.6	com.woty.android.model.MultiOptionChallenge	58
5.3.7	com.woty.android.model.QRChallenge	59
5.3.8	com.woty.android.model.Description	59
5.3.9	com.woty.android.model.TextDescription	60
5.3.10	com.woty.android.model.ImageDescription	61
5.3.11	com.woty.android.model.Quest	62
5.3.12	com.woty.android.model.QuestHistory	63
5.3.13	com.woty.android.model.Rank	64
5.3.14	com.woty.android.model.Session	66
5.4	Specifiche componente View	69
5.4.1	com.woty.android.view.Vista	69
5.4.2	com.woty.android.view.Login	70
5.4.3	com.woty.android.view.Home	71

5.4.4	com.woty.android.view.ModProfile	72
5.4.5	com.woty.android.view.PendingQuests	73
5.4.6	com.woty.android.view.SolveQuest	74
5.4.7	com.woty.android.view.Ranks	76
5.4.8	com.woty.android.view.PlayerProfile	78
5.4.9	com.woty.android.view.Help	79
5.4.10	com.woty.android.view.About	80
5.5	Specifica componente Presenter	82
5.5.1	com.woty.android.presenter.Presenter	82
5.5.2	com.woty.android.presenter.LoginP	83
5.5.3	com.woty.android.presenter.HomeP	83
5.5.4	com.woty.android.presenter.PlayerProfileP	84
5.5.5	com.woty.android.presenter.ModProfileP	85
5.5.6	com.woty.android.presenter.PendingQuestsP	86
5.5.7	com.woty.android.presenter.SolveQuestP	87
5.5.8	com.woty.android.presenter.RanksP	88
5.5.9	com.woty.android.presenter.C2dmMessageReceiver	89
5.5.10	com.woty.android.presenter.C2dmRegistrationReceiver	89
5.6	Specifica componente Util	91
5.6.1	com.woty.android.util.JParser	91
5.7	Specifica package exceptions	92
5.7.1	com.woty.android.exceptions.NotLoggedException	92
5.7.2	com.woty.android.exceptions.InvalidArgumentException	92

Elenco delle tabelle

1	Mappa metodo-funzione per SessionsController	33
2	Mappa permessi-metodi	34

Elenco delle figure

1	Desktop - Diagramma Componenti	37
2	Desktop - Diagramma Classi	38
3	Desktop - Classe Manager	39
4	Desktop - Classe Exception	42
5	Desktop - Classe Client	43
6	Desktop - libreria libMaia	44
7	Desktop - Classe LoginWindow	45
8	Desktop - Classe Account	46
9	Desktop - libreria SimpleCrypt	48
10	Desktop - Classe Preferences	48
11	Mobile - Diagramma Componenti	51
12	Mobile - Diagramma Classi	52
13	Classe com.woty.android.model.Resource	53
14	Classe com.woty.android.model.User	54
15	Classe com.woty.android.model.Challenge	56
16	Classe com.woty.android.model.TrueFalseChallenge	57
17	Classe com.woty.android.model.ComboChallenge	58
18	Classe com.woty.android.model.MultiOptionChallenge	58
19	Classe com.woty.android.model.QRChallenge	59
20	Classe com.woty.android.model.Description	59
21	Classe com.woty.android.model.TextDescription	60
22	Classe com.woty.android.model.ImageDescription	61
23	Classe com.woty.android.model.Quest	62
24	Classe com.woty.android.model.QuestHistory	63

25	Classe com.woty.android.model.Rank	64
26	Classe com.woty.android.model.Session	66
27	Classe com.woty.android.view.Vista	69
28	Classe com.woty.android.view.Login	70
29	Classe com.woty.android.view.Home	71
30	Classe com.woty.android.view.ModProfile	72
31	Classe com.woty.android.view.PendingQuests	73
32	Classe com.woty.android.view.SolveQuest	74
33	Classe com.woty.android.view.Ranks	76
34	Classe com.woty.android.view.PlayerProfile	78
35	Classe com.woty.android.view.Help	79
36	Classe com.woty.android.view.About	80
37	Classe com.woty.android.presenter.Presenter	82
38	Classe com.woty.android.presenter.LoginP	83
39	Classe com.woty.android.presenter.HomeP	83
40	Classe com.woty.android.presenter.PlayerProfileP	84
41	Classe com.woty.android.presenter.ModProfileP	85
42	Classe com.woty.android.presenter.PendingQuestsP	86
43	Classe com.woty.android.presenter.SolveQuestP	87
44	Classe com.woty.android.presenter.RanksP	88
45	Classe com.woty.android.presenter.C2dmMessageReceiver	89
46	Classe com.woty.android.presenter.C2dmRegistrationReceiver	89
47	Classe com.woty.android.remote.JParser	91
48	Classe com.woty.android.exceptions.NotLoggedException	92
49	Classe com.woty.android.exceptions.InvalidArgumentException	92

1 Introduzione

1.1 Scopo del documento

Con il presente documento si vuole dare una definizione dettagliata dell'architettura del sistema Woty attraverso un approccio top-down. Di ogni sottosistema infatti verrà presentata dapprima la struttura dei componenti, poi verranno specificate le classi, ed infine di ogni classe si indicheranno metodi e campi dati.

1.2 Scopo del prodotto

Il sistema software PMAC si pone come obiettivo la realizzazione di una piattaforma innovativa per l'apprendimento comportamentale nell'ambito della sicurezza del lavoro, che utilizzi le tecniche della gamification per incentivare il coinvolgimento e la partecipazione degli utenti e per scardinare l'instaurarsi di abitudini errate.

1.3 Glossario

Per evitare ridondanze tutti i termini e gli acronimi presenti nel seguente documento che necessitano di definizione saranno seguiti da una "g" ad apice (E.g. User^g) alla loro prima occorrenza e saranno riportati in un documento esterno denominato *Glossario.pdf*. Tale documento accompagna e completa il presente e consiste in un listato ordinato di termini e acronimi con le rispettive spiegazioni.

1.4 Riferimenti

1.4.1 Normativi

- Norme generali del progetto:
vedi documento fornito in allegato *NormeDiProgetto_v3.pdf*
- Capitolato d'appalto:
<http://www.math.unipd.it/~tullio/IS-1/2011/Progetto/C3.pdf>
- Analisi dei Requisiti:
vedi documento fornito in allegato *AnalisiDeiRequisiti_v3.pdf*
- Lucidi delle lezioni del corso di "Ingegneria del Software" mod. A e B del Prof.re Riccardo Cardin Vardanega:
<http://www.math.unipd.it/~tullio/IS-1/2011/Dispense>
<http://www.math.unipd.it/~rcardin/sweb.html>

1.4.2 Informativi

- Associazioni tra modelli:
<http://api.rubyonrails.org/classes/ActiveRecord/Associations/ClassMethods.html>

2 Standard di progetto

2.1 Standard di progettazione architetturale

Questo argomento è trattato nel documento di Specifica Tecnica, alla versione 2.0 (allegato *SpecificaTecnica_v2.pdf*).

2.2 Standard di documentazione del codice

Per le modalità di documentazione del codice si faccia riferimento al documento Norme di progetto, alla versione 3.0 (allegato *NormeDiProgetto_v3.pdf*).

2.3 Standard di denominazione di entità e relazioni

Precise indicazioni per la nomenclatura di entità e relazioni sono descritte nel documento Norme di Progetto, alla versione 3.0 (allegato *NormeDiProgetto_v3.pdf*).

2.4 Standard di programmazione

Descrizioni e norme per la codifica si trovano nel documento Norme di Progetto, alla versione 3.0 (allegato *NormeDiProgetto_v3.pdf*).

2.5 Strumenti di lavoro

Riferimenti e specifiche agli strumenti di lavoro si trovano nel documento Norme di Progetto, alla versione 3.0 (allegato *NormeDiProgetto_v3.pdf*).

3 Specifica Woty Server

3.1 Introduzione

Verrà di seguito presentata l'architettura di dettaglio del sottosistema server, analizzando ogni risorsa presente con una descrizione dei suoi attributi e dei suoi metodi, oltre a relazioni con altre classi/componenti.

Come anticipato nel documento di Specifica Tecnica, questo sottosistema implementa il design pattern Model-View-Controller. Per rendere uniforme la struttura del documento, la descrizione di dettaglio sarà orientata alla risorsa: per ognuna di esse saranno descritte le componenti coinvolte al suo corretto funzionamento. La maggior parte risponde in maniera standardizzata, di conseguenza il presente documento fornisce un'implementazione standard, che verrà poi riferita nelle descrizioni concrete delle risorse che non necessitano di adattamenti particolari.

3.2 Wildcard

Verranno utilizzate le seguenti wildcard nel capitolo corrente con riferimento la risorsa in questione:

- *model* riferimento al nome singolare della risorsa.
- *models* riferimento al nome plurale della risorsa.

3.3 Proprietà del linguaggio Ruby

Di seguito elencate le proprietà del linguaggio notevoli adottate anche per la scrittura della presente documentazione

- Metodi di classe identificati come *self.nome_metho*
- Omessi tipi di ritorno e dei parametri nella segnatura dei metodi

3.4 Implementazione standard di una risorsa

Di seguito descritte in dettaglio le componenti coinvolte nel funzionamento standard di una risorsa.

3.4.1 Model

- Classe derivata da ActiveRecord;
- File contenitore denominato *Model.rb* e situato in /app/models
- Definisce gli attributi e la loro accessibilità;
- Definisce eventuali associazioni e dipendenze da altri modelli a livello database;
- Definisce le validazioni (se esistono) per ognuno dei campi dati;
- Definisce metodi ricorrenti *inerenti alla risorsa* per l'estrazione di dati;
- Definisce eventuali *callback^g* necessari al funzionamento;

Attributi dei modelli L'implementazione effettiva dell'accesso a campi dati d'istanza dei modelli sul framework utilizzato non fa uso di attributi di istanza del linguaggio ruby (usa le colonne della tabella su database per generare metodi). Tuttavia, ai fini di documentazione, riteniamo più utile trattarli come tali per facilitare la lettura. Di conseguenza saranno inclusi e denominati attributi, pur essendo consapevoli che in realtà non sono tali.

Descrizione validazioni per attributi dei modelli La classe *ActiveRecord::Base* prevede di definire a livello applicazione i vincoli degli attributi per la validazione di un’istanza di un modello. Per rapidità descrittiva saranno incluse nelle singole risorse le validazioni come definito nel riferimento informativo *Validazione modelli*. Riteniamo importante l’inclusione dei tali all’interno del documento. Verranno utilizzati i seguenti tag nella descrizione degli attributi per identificare rapidamente le validazioni:

Descrizione associazioni tra modelli La classe *ActiveRecord::Base* prevede di definire a livello applicazione le tipologie di associazioni per la creazione dinamica di metodi per l’accesso a risorse associate. Per rapidità descrittiva saranno incluse nelle singole risorse le associazioni come definito nel riferimento informativo *Associazioni tra modelli*. Riteniamo importante l’inclusione dei tali all’interno del documento.

- #REQUIRED [Attributo richiesto]
- #UNIQUE [Attributo unico]
- #AUTO [Attributo automaticamente generato]
- #EXP [Attributo conforme all’espressione EXP]

Descrizione associazioni tra modelli La classe *ActiveRecord::Base* prevede di definire a livello applicazione le tipologie di associazioni per la creazione dinamica di metodi per l’accesso a risorse associate. Per rapidità descrittiva saranno incluse nelle singole risorse le associazioni come definito nel riferimento informativo *Associazioni tra modelli*. Riteniamo importante l’inclusione dei tali all’interno del documento in quanto i metodi generati sono frequentemente utilizzati.

3.4.2 Controller

L’implementazione standard di un controller soddisfa le seguenti:

- Classe derivata da ApplicationController;
- Risponde a richieste di tipo html, json.
- Filtra o nega l’accesso alle risorse non accessibili dall’utente corrente.
- Specializza i 7 metodi dell’interfaccia Resource;
- Definisce eventuali *callback^g* necessari al funzionamento;

3.4.3 View

L’implementazione standard delle viste definisce sei file necessari alla renderizzazione. Di questi, due sono elementi parziali da includere da altre parti. All’interno delle viste è gestita l’autorizzazione alla lettura dei dati. Le viste non sono classi e di conseguenza l’implementazione del pattern MVC è leggermente diversa, più centralizzata rispetto al controller (non è implementato il design pattern observer).

_model.html.erb Definizione di renderizzazione risorsa.

_form.html.erb Definizione di renderizzazione di un form html per la creazione/modifica di una risorsa.

new.html.erb Definizione di interfaccia per la creazione di una nuova risorsa (includendo l’html generato da _form.html.erb)

edit.html.erb Definizione di interfaccia per la modifica di una nuova risorsa (includendo l’html generato da _form.html.erb)

index.html.erb Definizione di interfaccia per un listato delle risorse disponibili.

show.html.erb Definizione di interfaccia per l'accesso e visualizzazione a una risorsa (incluendo l'html generato da `_model.html.erb`)

3.4.4 Fogli di stile

Situati in `/assets/stylesheets`, contengono gli stili necessari alle view della risorsa.

3.4.5 Javascript

Situati in `/assets/javascripts`, contengono il javascript necessario alle view della risorsa.

3.5 Comportamenti e necessità anomale

Le necessità e comportamenti anomali saranno descritti specificamente per le necessità particolari della risorsa in oggetto. Di seguito un linea di guida utile alla comprensione successiva.

3.5.1 Decorator

Il design pattern è fondamentalmente utilizzato per il miglioramento della qualità del codice. Le funzionalità aggiunte sono solamente metodi puramente grafici e servono a mantenere facilmente leggibile il codice delle viste. L'implementazione del pattern è carico di una libreria esterna.

- Classe derivata da `Draper::Decorator`;
- Denominata `ModelDecorator`;
- Contiene tutti i metodi necessari alla pulizia del markup delle viste (se necessari);
- Contiene eventuali metodi di classe relativi al modello non conformi alle norme di codifica per i modelli (troppo generici);

3.5.2 Helper

Gli helper sono una caratteristica del framework RoR, che preferiamo usare al minimo possibile e per compiti basilari a causa della non riutilizzabilità del codice (non sono metodi di classe).

- Modulo denominato `ModelsHelper`;
- Incluso automaticamente per l'utilizzo dei metodi nelle viste;
- Definisce i metodi di utilità necessari per piccole elaborazioni;
- Usato al minimo necessario (Non OOP);

3.5.3 Gerarchie standard

Di seguito l'implementazione del design pattern Single Table Inheritance per le gerarchie di risorse.

Classe base E' definito un modello standard dichiarante un campo `type`.

Classi derivate E' definito un modello standard derivato dalla classe base (non da ActiveRecord).

3.6 Specifiche risorse framework

Di seguito le componenti framework necessarie alle risorse

3.6.1 Classe ApplicationController

Classe derivata da *ActionController::Base*. La classe base garantisce le principali operazioni comuni a tutti i controller, in particolare la costruzione di risposte http.

Funzione della classe La classe serve a definire metodi, callback e impostazioni comuni a tutti i controller delle risorse.

Metodi

+ `logged_user`

Ritorna l'utente della sessione corrente.

+ `user_is_logged?`

Ritorna *true* se la sessione corrente corrisponde a un utente iscritto, *false* altrimenti.

+ `normalize_format`

Se non specificato un formato di richiesta nella chiamata http, lo normalizza su html.

+ `previous_page`

ritorna un url relativo alla pagina precedente se specificata nella richiesta http, altrimenti l'url della homepage

+ `current_ability`

Ritorna un'istanza della classe Ability definita per l'utente connesso

Callbacks

load_and_authorize_resource Definisce prima di ogni chiamata a metodo un callback che se necessario istanzia un attributo ed eventualmente mostra una pagina di blocco per azioni non accessibili all'utente corrente.

normalize_format Definisce prima di ogni chiamata a metodo l'esecuzione della procedura corrispondente al metodo omonimo.

3.6.2 Classe ActiveRecord::Base

La classe rappresenta l'implementazione del design pattern omonimo. Garantisce i metodi basilari per l'esecuzione dei metodi CRUD su una risorsa situata all'interno di un database relazionale. Gestisce le associazioni tramite la creazione di metodi a runtime per l'accesso alle entità associate. Gestisce le dipendenze tra le associazioni.

Funzione della classe Classe astratta alla base della gerarchia di tutte le classi modello.

Attributi

- `integer id`

Rappresenta l'id univoco di una generica istanza di classe di modello.

- `datetime created_at`

Data di creazione di un'istanza della classe, generata automaticamente.

- `datetime updated_at`

Data di ultima modifica.

3.7 Specifiche risorse standard

3.7.1 User

Funzione della risorsa User è la risorsa che modella gli utenti registrati al sistema. E' la classe base della gerarchia degli utenti. La gerarchia è utilizzata per la creazione delle sessioni e l'autorizzazione. Rispetta l'implementazione standard descritta precedentemente. Di seguito verranno elencate le caratteristiche specifiche delle classi coinvolte.

Modello

Attributi

+ `email #REQUIRED #UNIQUE`

Rappresenta l'e-mail associata ad ogni User, necessaria per il login

- `password #REQUIRED`

La password dell'utente. Per questioni di sicurezza nel DB viene salvato un hash della stessa nella colonna `password_digest`

+ `locale`

Identificativo della lingua dello User

+ `name #REQUIRED`

Nome

+ `surname #REQUIRED`

Cognome

+ `type #REQUIRED`

Il tipo di User. Colonna necessaria per attivare il pattern Single Table Inheritance.

+ `workgroup_id #REQUIRED`

Id del Workgroup cui appartiene l'User

+ `avatar`

Avatar dello User

+ `exp`

Punti esperienza dello User

+ `status`

Stato personale dello User

Associazioni Le seguenti generano dinamicamente metodi per la classe come descritto nel riferimento informativo *Associazioni tra modelli*

- `belongs_to workgroup`
- `has_one customer through workgroup`
- `has_many completed_achievements`
- `has_many achievements through completed_achievements`
- `has_many completed_achievement_element through completed_achievements`
- `has_many stream_elements`
- `has_many comments`
- `has_many likes`

- has_one signup_element
- has_many completed_achievement_elements
- has_many status_change_elements
- has_many quest_histories
- has_many quests through quest_histories
- has_many pending_quests through quest_histories

Metodi

+ `experience`

Ritorna il valore derivante dal calcolo dell'esperienza ottenuta dall'utente.

+ `experience_to_next_level`

Ritorna i punti mancanti all'avanzamento di livello.

+ `admin?`

True se l'utente corrente è un admin, false altrimenti.

+ `self.subclasses_strings`

ritorna un url relativo alla pagina precedente se specificata nella richiesta http, altrimenti l'url della homepage.

+ `current_ability`

Ritorna un'istanza della classe Ability definita per l'utente connesso.

+ `self.roles`

Ritorna il nome della classe formattato per l'autorizzazione.

+ `level`

Ritorna il livello dell'utente.

+ `points`

Ritorna il punteggio dell'utente.

Moduli inclusi

- *Achievable::User*
- *Stats::User*

Controller Implementazione standard di risorsa.

View Implementazione standard di risorsa.

Decorator

+ `linked_namesurname`

Ritorna il codice html necessario per un link al profilo di un utente specifico con maschera nome e cognome.

+ `linked_name_surname_with_icon`

Ritorna il codice html necessario per un link al profilo di un utente specifico con maschera nome e cognome e un'icona se l'utente è un admin.

+ `namesurname`

Ritorna il nome e cognome dell'utente.

+ avatar

Ritorna l'html necessario alla visualizzazione dell'avatar.

+ thumbnail

Ritorna l'html necessario a un avatar di piccole dimensioni.

+ mini_thumbnail

Ritorna l'html necessario a un avatar di dimensioni minime.

+ level_bar

Ritorna l'html necessario alla visualizzazione di una barra con il livello.

+ self.most_active

Ritorna l'utente più attivo in termini sociali

+ self.average_recent_streams

Ritorna la media di stream recenti visualizzati dagli utenti

+ self.average_comments

Ritorna la media dei commenti per utente

+ self.average_likes

Ritorna la media dei like per utente

+ self.average_level

Ritorna il livello medio ottenuto dagli utenti

+ self.average_quests

Ritorna la media delle quest completate per utente.

3.7.2 SuperUser

Funzione della risorsa Specializzazione della classe base User. Implementazione standard.

Modello

Attributi Solamente quelli ereditati da User

Associazioni Solamente quelle ereditate da User

Metodi Solamente quelli ereditati da User

Moduli inclusi Solamente quelli ereditati da User

Controller Implementazione standard di risorsa.

View Implementazione standard di risorsa.

3.7.3 DesktopUser

Funzione della risorsa Specializzazione della classe base User. Implementazione standard.

Modello

Attributi Solamente quelli ereditati da User

Associazioni Solamente quelle ereditate da User

Metodi Solamente quelli ereditati da User

Moduli inclusi Solamente quelli ereditati da User

Controller Implementazione standard di risorsa.

View Implementazione standard di risorsa.

3.7.4 MobileUser

Funzione della risorsa Specializzazione della classe base User. Implementazione standard.

Modello

Attributi

+ `reg_id`

E' l'id di registrazione a Google C2DM del dispositivo Android associato al Mobile User. Viene aggiornato ad ogni login dall'applicazione mobile, e settato a nil al logout.

+ `token`

Token di sicurezza usato dall'applicazione Android per l'autenticazione. Viene creato ad ogni login (dell'applicazione mobile) dal controller Xmlrpc

Associazioni Solamente quelle ereditate da User

Metodi Solamente quelli ereditati da User

Moduli inclusi Solamente quelli ereditati da User

Controller Implementazione standard di risorsa.

View Implementazione standard di risorsa.

3.7.5 Admin

Funzione della risorsa Specializzazione della classe base User. Implementazione standard.

Modello

Attributi Solamente quelli ereditati da User

Associazioni Solamente quelle ereditate da User

Metodi Solamente quelli ereditati da User

Moduli inclusi Solamente quelli ereditati da User

Controller Implementazione standard di risorsa.

View Implementazione standard di risorsa.

3.7.6 Customer

Funzione della risorsa Customer è la risorsa che modella i clienti del fornitore del servizio. Ogni customer ha vari workgroup che a loro volta contengono una serie di user. Ogni customer inoltre sottoscrive un plan, che definisce il livello di servizio per il customer. I costi quindi, sono calcolati tra questa classe e la classe Plan. Nel sistema è un'entità molto importante in quanto tutti gli utenti fruiscono dei contenuti all'interno del loro customer. Di seguito verranno elencate le caratteristiche specifiche delle classi coinvolte.

Modello

Attributi

- + `email #REQUIRED #UNIQUE`
L'e-mail associata al Customer
- + `name #REQUIRED #UNIQUE`
Ragione sociale del Customer
- + `plan_id #REQUIRED #UNIQUE`
Id del Plan (tariffa) associato al Customer
- + `max_accounts #REQUIRED`
Numero massimo di licenze disponibili per il Customer

Associazioni Le seguenti generano dinamicamente metodi per la classe come descritto nel riferimento informativo *Associazioni tra modelli*

- `belongs_to plan`
- `has_many users through workgroups`
- `has_many stream_elements`
- `has_one superuser through users`

Metodi

- + `unique_superuser`
False se il customer corrente ha più di un superuser, true altrimenti.
- + `remaining_accounts`
Ritorna la differenza tra `max_accounts` e gli user attualmente istanziati dal customer

Controller Implementazione standard di risorsa.

View Implementazione standard di risorsa.

Decorator

- + `linked_name`
Ritorna il codice html necessario per un link al customer con maschera nome.
- + `linked_name_surname_with_icon`
Ritorna il codice html necessario per un link al profilo di un utente specifico con maschera nome e cognome e un'icona se l'utente è un admin.
- + `self.average_income`
Ritorna la media dei guadagni per cliente
- + `self.preferred_plan`
Ritorna il plan con più preferenze rispetto ai customer

3.7.7 WorkGroup

Funzione della classe Rappresenta un reparto di un'azienda cliente. I reparti sono composti dagli utenti facenti parte. Gli utenti in reparti lavorano anche in team, e sono disponibili classifiche di reparto. Sono inoltre utilizzati per l'assegnazione delle quest, in quanto il reparto è una delle caratteristiche per le quali sono influenzate le categorie di rischio.

Attributi

- **wgcategory_id**
Identificativo della categoria di appartenenza del WorkGroup
- **name**
Nome assegnato al WorkGroup.
- **customer_id**
Identificativo dell'azienda/cliente di appartenenza del WorkGroup.
- **name**
Nome del WorkGroup.

Associazioni Le seguenti generano dinamicamente metodi per la classe come descritto nel riferimento informativo *Associazioni tra modelli*

- belongs_to wgcategory
- belongs_to customer
- has_many users
- has_many custom_quest
- has_many quest_association
- has_many quest_categories through quest_associations
- has_many quests through quest_categories

Controller Implementazione standard di risorsa.

View Implementazione standard di risorsa.

Decorator

- + **linked_members_list**
Ritorna il codice html necessario per una lista linkata degli utenti del workgroup
- + **linked_name**
Ritorna il codice html necessario per il nome del workgroup con un link alla sua pagina.

3.7.8 Plan

Funzione della risorsa Plan è la risorsa che modella le formule d'acquisto del prodotto. Il paradigma Software as a Service uniforma nel miglior modo possibile il maggior numero di clienti. Per il nostro caso, possono essere sottoscritti vari piani di acquisto, e nel caso in cui il cliente necessiti di caratteristiche personali è possibile assegnare dei plan privati. All'interno del plan sono definite alcune abilità che possono o meno essere utilizzate a seconda del prezzo pagato. Di seguito verranno listate le caratteristiche specifiche delle classi coinvolte.

Modello

Attributi

- + `name #REQUIRED #UNIQUE`
Nome del Plan
- + `max_custom_quests #REQUIRED #>0`
Numero massimo di Quest personalizzabili assegnabili ad un Customer che scelga il Plan
- + `videoquests_are_allowed`
True se le Quest Video sono disponibili per i Customer, false altrimenti
- + `mobiles_are_allowed`
True se il Plan prevede MobileUser, false altrimenti
- + `description #REQUIRED`
Descrizione testuale del Plan
- + `price_per_account #REQUIRED #>=0`
Prezzo per licenza
- + `is_public`
True se il Plan risulta visibile pubblicamente (alla pagina /pricing), false altrimenti

Associazioni Le seguenti generano dinamicamente metodi per la classe come descritto nel riferimento informativo *Associazioni tra modelli*

- has_many customers

Metodi Nessuno

Controller Implementazione standard di risorsa.

View Implementazione standard di risorsa.

Decorator

- + `public_link`
Ritorna il codice html necessario per un link alla pagina pubblica contenente i piani visibili
- + `active_customer_list`
Ritorna il codice html necessario per la lista di sottoscrittori del piano, linkati.
- + `self.statistics`
Ritorna una stringa indicante il numero di piani visibili pubblicamente

3.7.9 Quest

Funzione della classe La classe modella le quest e la loro composizione. Di fatto le quest sono composte di una descrizione e una sfida: entrambe possono essere di vario tipo e sono rappresentate tramite gerarchie. Le quest inoltre hanno una categoria che può essere inclusa in un workgroup: gli utenti facenti parte sono in grado di svolgere le quest con le categorie associate.

Associazioni Le seguenti generano dinamicamente metodi per la classe come descritto nel riferimento informativo *Associazioni tra modelli*

- belongs_to quest_challenge
- belongs_to quest_description
- belongs_to quest_category
- has_many quest_histories
- has_many users through quest_histories

Attributi

+ granted_exp

Rappresenta l'esperienza guadagnata nel caso di corretta risoluzione della Quest.

+ quest_challenge_id

Identificativo della sezione QuestChallenge che compone la Quest.

+ quest_description_id

Identificativo della sezione QuestDescription che compone la Quest.

Metodi

+ correct?(ans)

true se la quest è corretta per la risposta passata con il parametro

+ self.periodical_assign

Crea e assegna periodicamente una quest a determinati User.

+ quest_challenge_type

Ritorna il tipo della sfida associata alla quest

+ quest_description_type

Ritorna il tipo della descrizione associata alla quest

+ self.subclasses_strings

Ritorna un vettore di stringhe corrispondenti ai nomi delle sottoclassi

Controller Implementazione standard di risorsa.

View Implementazione standard di risorsa.

Helper

+ pending_quests_for(user)

Ritorna l'html necessario a un link per mostrare il numero di quest assegnate e ancora da svolgere.

3.7.10 QuestDescription

Funzione della risorsa QuestDescription è la risorsa che modella gli utenti registrati al sistema. È la classe base della gerarchia delle tipologie di descrizione di una quest. La gerarchia è utilizzata per la composizione di una quest. Rispetta l'implementazione standard per gerarchie descritta precedentemente. Di seguito verranno listate le caratteristiche specifiche delle classi coinvolte.

Modello

Attributi

+ name #REQUIRED

Associazioni Le seguenti generano dinamicamente metodi per la classe come descritto nel riferimento informativo *Associazioni tra modelli*

- has_many quests

Metodi Nessuno

Controller Implementazione standard di risorsa.

View Implementazione standard di risorsa.

3.7.11 TextDescription

Funzione della risorsa La classe modella una QuestDescription che contiene un testo.

Modello

Attributi

+ text #REQUIRED

Contiene il testo da visualizzare

Associazioni Solamente quelle ereditate da QuestDescription

Metodi Nessuno

Controller Implementazione standard di risorsa.

View Implementazione standard di risorsa.

3.7.12 ImageDescription

Funzione della risorsa La classe modella una QuestDescription che contiene un'immagine.

Modello

Attributi

+ img #REQUIRED

Contiene l'immagine da visualizzare

Associazioni Solamente quelle ereditate da QuestDescription

Metodi Nessuno

Controller Implementazione standard di risorsa.

View Implementazione standard di risorsa.

3.7.13 VideoDescription

Funzione della risorsa La classe modella una QuestDescription che contiene un video.

Modello

Attributi

+ `video #REQUIRED #UNIQUE`
Contiene il video da visualizzare

Associazioni Solo quelle ereditate da QuestDescription

Metodi Nessuno

Controller Implementazione standard di risorsa.

View Implementazione standard di risorsa.

3.7.14 QuestChallenge

Funzione della classe Quest Challenge è la classe base per le sfide delle quest. Rappresenta il componente sfida delle quest che può essere di varie tipologie, definite tramite le sottoclassi.

Modello

Attributi

+ `question #REQUIRED`
Contiene il quesito rivolto all'utente sotto forma di stringa.
- `type #REQUIRED`
Specializza il tipo di QuestChallenge. Necessario per il triggering del pattern STI
+ `name #REQUIRED #UNIQUE`
Nome relativo alla QuestChallenge.

Associazioni Le seguenti generano dinamicamente metodi per la classe come descritto nel riferimento informativo *Associazioni tra modelli*

- `has_many quests`

Metodi

+ `tfstatements_must_not_have_empty_values`
Metodo per validazione attributo tfstatements
+ `tfstatements_must_be_boolean`
Metodo per validazione attributo tfstatements

Controller Implementazione standard di risorsa.

View Implementazione standard di risorsa.

3.7.15 ComboChallenge

Funzione della classe Rappresenta un quesito a risposta multipla con una ed una sola risposta corretta.

Modello

Attributi

+ `rstatement #REQUIRED #>0`

Identifica la risposta corretta

- `tfstatements #REQUIRED`

Contiene la serializzazione YAML dell'hash di risposte possibili. Ciò è reso possibile da ActiveRecord.

Associazioni Solo quelle ereditate da QuestChallenge

Metodi

+ `rstatement_must_be_valid()`

Controlla se il campo rstatement ha un valore legale.

+ `correct(String)`

Restituisce true se l'hash di risposte ans è valido per la challenge corrente, false altrimenti

Controller Implementazione standard di risorsa.

View Implementazione standard di risorsa.

3.7.16 TrueFalseChallenge

Funzione della risorsa La classe modella una tipologia particolare di QuestChallenge contenente più affermazioni, ognuna delle quali può essere vera o falsa.

Modello

Attributi

+ `tfstatements #REQUIRED`

Tfstatements contiene la serializzazione YAML dell'hash relativo alle n affermazioni e ai rispettivi valori.

Associazioni Solo quelle ereditate da QuestChallenge

Metodi

+ `correct?(ans)`

Restituisce true se l'hash di risposte ans è valido per la challenge corrente, false altrimenti

Controller Implementazione standard di risorsa.

View Implementazione standard di risorsa.

3.7.17 MultiOptionChallenge

Funzione della risorsa La classe modella una tipologia particolare di QuestChallenge contenente più affermazioni, ognuna delle quali può essere valida o meno. Si differenzia da TrueFalseChallenge a livello logico, in quanto più che possibili affermazioni che possono essere vere e false, modella possibili opzioni che possono essere valide o meno.

Modello

Attributi

+ `tfstatements #REQUIRED`

Tfstatements contiene la serializzazione YAML dell'hash relativo alle n affermazioni e ai rispettivi valori.

Associazioni Solo quelle ereditate da QuestChallenge

Metodi

+ `correct?(ans)`

Restituisce true se l'hash di risposte ans è valido per la challenge corrente, false altrimenti

Controller Implementazione standard di risorsa.

View Implementazione standard di risorsa.

3.7.18 Achievement

Funzione della risorsa Gli achievement sono modellati tramite una serie di condizioni legate dall'operatore logico AND. Per semplicità di implementazione non si è ritenuta necessaria l'implementazione di altri operatori logici. La validità di un achievement è calcolata per uno user e valida tutte le condizioni associate.

Modello

Attributi

+ `name #REQUIRED #UNIQUE`

Nome identificativo dell'achievement

- `description #REQUIRED`

Descrizione dell'achievement

Associazioni Le seguenti generano dinamicamente metodi per la classe come descritto nel riferimento informativo *Associazioni tra modelli*

- `has_many conditions`
- `has_many completed_achievements`
- `has_many users through completed_achievements`

Metodi

+ `check_and_award(user)`

Crea un nuovo CompletedAchievement per lo user se l'achievement è completato.

+ `valid_for?(user)`

True se l'achievement è completato per lo user, false altrimenti.

Controller Implementazione standard di risorsa.

View Implementazione standard di risorsa.

Decorator

- + `linked_name`
Ritorna il codice html necessario per un link all'achievement con maschera nome.
- + `linked_label`
Ritorna il codice html necessario per una label con testo il nome dell'achievement e un link alla sua pagina.
- + `linked_achievers_list`
Ritorna il codice html necessario per una lista con link degli utenti che hanno completato l'achievement

3.7.19 Like

Funzione della risorsa Like è la risorsa che modella i like effettuati dagli utenti. Attualmente è possibile effettuare i like solo su elementi membri della gerarchia StreamElement.

Modello

Attributi

- + `user_id #REQUIRED`
Lo user che effettua il like
- `stream_element_id #REQUIRED`
L'elemento sul quale è stato effettuato il like

Associazioni Le seguenti generano dinamicamente metodi per la classe come descritto nel riferimento informativo *Associazioni tra modelli*

- belongs_to user
- belongs_to stream_element
- has_one customer through stream_element

Metodi Nessuno

Controller Implementazione standard di risorsa.

View Implementazione standard di risorsa.

3.7.20 Comment

Funzione della risorsa Comment è la risorsa che modella i commenti effettuati dagli utenti. Ad oggi è possibile commentare solamente elementi membri della gerarchia StreamElement.

Modello

Attributi

- + `user_id #REQUIRED`
Utente che effettua il commento
- `stream_element_id #REQUIRED`
StreamElement commentato
- + `body #REQUIRED`
Corpo del commento

Associazioni Le seguenti generano dinamicamente metodi per la classe come descritto nel riferimento informativo *Associazioni tra modelli*

- belongs_to user
- belongs_to stream_element
- has_one customer through stream_element

Metodi Nessuno

Controller Implementazione standard di risorsa.

View Implementazione standard di risorsa.

3.7.21 StreamElement

Funzione della risorsa StreamElement è la classe base per gli elementi di attività visibili nel pannello utente. Le attività possono essere di vario genere, permettono l’interazione tramite like e commenti, e costituiscono il principale elemento social del sistema.

Modello

Attributi

+ `type #REQUIRED #UNIQUE`

Tipo di StreamElement per Single Table Inheritance

Associazioni Le seguenti generano dinamicamente metodi per la classe come descritto nel riferimento informativo *Associazioni tra modelli*

- belongs_to user
- has_one customer through user
- has_many comments
- has_many likes

Metodi

+ `self.subklasses`

Ritorna un array di oggetti di tipo *class* contenente le sottoclassi.

+ `owner`

Ritorna un’istanza di UserDecorator che decora il proprietario dello StreamElement

+ `title`

Ritorna un titolo per lo StreamElement

+ `owner`

Ritorna una descrizione per lo StreamElement

Controller Implementazione standard di risorsa.

View Implementazione standard di risorsa.

Esiste inoltre un elemento parziale per la generazione di una versione più piccola dello StreamElement.

3.7.22 Specifica Classe QuestHistory

Funzione della risorsa Una QuestHistory relaziona uno User con una Quest. Viene creata quando uno User svolge una Quest oppure quando il sistema assegna ad uno User una Quest.

Modello

Attributi

+ `quest_id #REQUIRED`

Identifica la Quest relativa alla QuestHistory

+ `user_id #REQUIRED`

Identifica lo User relativo alla QuestHistory

- `done`

Viene settato a False in caso di assegnamento automatico di una Quest ad un User da parte del sistema, True altrimenti (lo User svolge una Quest)

- `successful`

True se la Quest è stata affrontata correttamente, False altrimenti

Associazioni Le seguenti generano dinamicamente metodi per la classe come descritto nel riferimento informativo *Associazioni tra modelli*

- `belongs_to quest`

- `belongs_to user`

Metodi

+ `self.clean undone(user_id, quest_id)`

Elimina le QuestHistory con user_id e quest_id passati come parametri e con attributo done settato a false. Viene richiamato quando uno User svolge una quest che gli era stata precedentemente assegnata.

+ `quest_name`

Ritorna il nome della Quest relativa alla QuestHistory.

+ `quest_name`

Ritorna l'exp ottenuta in caso di svolgimento corretto della Quest relativa alla QuestHistory

Controller Implementazione standard di risorsa.

View Implementazione standard di risorsa.

3.7.23 Ticket

Funzione della risorsa Rappresenta un Ticket creato da un SuperUser e rivolto al/agli Admin del sistema.

Modello

Attributi

- Derivati da ActiveRecord.

- **title**

Oggetto del Ticket

- **customer_id**

Identifica il Customer relativo al SuperUser che crea il Ticket

- **body**

Testo contenuto nel Ticket.

Associazioni Le seguenti generano dinamicamente metodi per la classe come descritto nel riferimento informativo *Associazioni tra modelli*

- belongs_to :customer
- has_one :superuser

Controller Implementazione standard di risorsa.

View Implementazione standard di risorsa.

3.7.24 CompletedAchievement

Funzione della risorsa Rappresenta un achievement ottenuto da un utente.

Modello

Attributi

- Derivati da ActiveRecord.

- + **user_id**

Id dell'utente che ha ottenuto l'achievement

- + **achievement_id**

Id dell'achievement ottenuto

Associazioni Le seguenti generano dinamicamente metodi per la classe come descritto nel riferimento informativo *Associazioni tra modelli*

- belongs_to achievement
- belongs_to user

Metodi

- + **stop_duplicates**

False se l'user ha già ottenuto l'achievement, true altrimenti.

- + **create_and_link_completed_achievement_element**

Crea un CompletedAchievementElement associato all'istanza

Controller Implementazione standard di risorsa.

View Implementazione standard di risorsa.

3.7.25 CompletedAchievementElement

Funzione della risorsa Specializzazione della classe StreamElement che rappresenta l'ottenimento di un achievement da parte di un utente.

Modello

Attributi

- Derivati da ActiveRecord.

+ `completed_achievement_element_id #REQUIRED`

Id del corrispondente CompletedAchievementElement

Associazioni Le seguenti generano dinamicamente metodi per la classe come descritto nel riferimento informativo *Associazioni tra modelli*

- belongs_to completed_achievement
- has_one achievement through completed_achievement

Metodi

+ `title`

Specializzazione dalla classe base

+ `description`

Specializzazione dalla classe base

Controller Implementazione standard di risorsa.

View Implementazione standard di risorsa.

3.7.26 Condition

Funzione della risorsa La classe modella una condizione necessaria al completamento di un achievement. Le condizioni sono composte da un operatore, uno dei valori definiti in *Achievable*, e un valore fissato dal creatore. Abbiamo deciso di limitare la libertà di creazione delle condizioni, in quanto altre soluzioni sarebbero state troppo complesse da implementare. Il sistema le rende comunque generabili dinamicamente.

Modello

Attributi

+ `property #REQUIRED`

Proprietà da controllare

+ `operator #REQUIRED`

Operator per l'espressione

+ `value #REQUIRED`

Valore definito dal creatore

Associazioni Le seguenti generano dinamicamente metodi per la classe come descritto nel riferimento informativo *Associazioni tra modelli*

- belongs_to achievement

Metodi

+ validates_for?(object)

Estrae la proprietà dall'object e la confronta con l'operatore e il valore fissato. True se la condizione risulta completata, false altrimenti.

Controller Implementazione standard di risorsa.

View Implementazione standard di risorsa.

3.7.27 CustomQuest

Funzione della classe Modella una Quest personalizzata, disponibile solo ad un determinato Workgroup

Relazioni con altre classi Derivata da Quest, è relazionata con Workgroup.

Associazioni Le seguenti generano dinamicamente metodi per la classe come descritto nel riferimento informativo *Associazioni tra modelli*

- belongs_to :workgroup

Attributi

+ workgroup_id

Identifica il Workgroup al quale la CustomQuest è associata

Controller Implementazione standard di risorsa.

View Implementazione standard di risorsa.

3.7.28 QuestAssociation

Funzione della classe Relaziona le QuestCategory con i Workgroup

Associazioni

- belongs_to :workgroup
- belongs_to :quest_category

Attributi

+ workgroup_id

Identifica il Workgroup al quale la QuestAssociation è relazionata

+ quest_category_id

Identifica la QuestCategory al quale la QuestAssociation è relazionata

Controller Implementazione standard di risorsa.

View Implementazione standard di risorsa.

3.7.29 QuestCategory

Funzione della classe Identifica una categoria di rischio al quale una SharedQuest è associata.

Associazioni

- has_many :quests
- has_many :quest_associations
- has_many :workgroups, :through => :quest_associations

Attributi**+ name**

Il nome testuale della QuestCategory

3.7.30 SharedQuest

Funzione della classe Rappresenta una Quest condivisa tra più Workgroup.

Associazioni

- belongs_to :quest_category
- has_many :workgroups, :through => :quest_associations

Attributi**+ quest_category_id**

Identifica la QuestCategory associata

3.7.31 SignupElement

Funzione della risorsa Specializzazione della classe StreamElement che rappresenta l'iscrizione di un utente al sistema.

Modello

Attributi Solo quelli ereditati da StreamElement

Associazioni Solo quelle ereditate da StreamElement

Metodi**+ title**

Specializzazione dalla classe base

+ description

Specializzazione dalla classe base

Controller Implementazione standard di risorsa.

View Implementazione standard di risorsa.

3.7.32 StatusChangeElement

Funzione della risorsa Specializzazione della classe StreamElement che rappresenta il cambio di stato di un utente.

Modello

Attributi

- Derivati da ActiveRecord.

+ `new_status #REQUIRED`

Nuovo stato dell'utente

Associazioni Solo quelle ereditate da StreamElement

Metodi

+ `title`

Specializzazione dalla classe base

+ `description`

Specializzazione dalla classe base

Controller Implementazione standard di risorsa.

View Implementazione standard di risorsa.

3.7.33 Wgcategorie

Funzione della classe Rappresenta una tipologia di Workgroup

Relazioni con altre classi Derivata da ActiveRecord, è relazionata con Workgroup.

Associazioni

- has_many :workgroups

Attributi

+ `name`

Il nome della tipologia di Workgroup

+ `description`

Descrizione testuale

3.8 Altre risorse

3.8.1 XmlrpcController

Il controller XmlrpcController, descritto nel file app/controllers/xmlrpc_controller.rb, contiene i metodi accessibili tramite richiesta XMLRPC. È stato deciso di usare questo sistema per tutte le richieste difficilmente mappabili come RESTful. Ciò è reso possibile grazie all'utilizzo della gemma rails-xmlrpc.

Metodi

+ `nQuest(email, password)`

Ritorna via XML il numero di Quest assegnate all'User caratterizzato dall'e-mail e password passate come parametri. Torna -1 in caso la doppia e-mail/password non corrisponda a nessun User. Viene utilizzato dall'applicazione Woty Desktop.

+ `getMobileToken(email, password)`

Controlla che la doppia e-mail/password passata come parametro corrisponda ad un MobileUser. In caso positivo genera un token mediante un algoritmo di criptazione AES a chiave segreta e lo ritorna incapsulato in XML. Altrimenti ritorna -1. Viene utilizzato dall'applicazione Woty Mobile.

+ getIdFromToken(token)

Controlla che il token passato sia effettivamente appartenente ad un MobileUser, del quale ritorna l'id incapsulato in XML. In caso negativo ritorna -1. Viene utilizzato dall'applicazione Woty Mobile in seguito ad un getMobileToken andato a buon fine. E' stato reso necessario un metodo a parte in quanto le librerie utilizzate dall'applicazione Woty Mobile per la comunicazione XMLRPC non supportano il ritorno di più parametri all'interno di uno stesso metodo.

3.8.2 Autenticazione

L'autenticazione è gestita da un controller non conforme a risorsa permanente. Il modello associato è infatti atipico in quanto non persistente su database. L'architettura rimane comunque fedele al pattern MVC.

Session (model)

Funzione della classe Garantisce ad un utente i permessi che gli spettano. I permessi sono stati descritti in maniera gerarchica.

Moduli inclusi

- *ActiveModel::Validations*
- *ActiveModel::Conversion*

Attributi

+ email #REQUIRED

Indirizzo email

+ password #REQUIRED

Password

+ id

Id utente

Metodi

+ initialize(attributes)

Costruisce l'oggetto con gli attributi parametro

+ correct_data

Controlla se la coppia email - password è corretta, se si imposta l'id corrispondente all'utente.

SessionsController (controller)

Funzione della classe Definisce i metodi necessari al funzionamento del sistema di autenticazione. Abbiamo mappato le azioni necessarie all'autenticazione in questa classe come definito dalla tabella *Mappa metodo-funzione per SessionsController*

Tabella 1: Mappa metodo-funzione per SessionsController

Metodo	Funzione
new	Pagina di login
create	Creazione di una sessione

(Continua alla pagina successiva)

(Continua dalla pagina precedente)

destroy	Distruzione di una sessione
edit, update, index, show	Nessuna

Attributi Nessuno

Metodi

+ **index**

Renderizza una risposta nulla.

+ **new**

Renderizza una pagina di login

+ **show**

Renderizza una risposta nulla.

+ **edit**

Renderizza una risposta nulla.

+ **create**

Crea una sessione attiva se i parametri sono corretti.

+ **update**

Renderizza una risposta nulla.

+ **destroy**

Distrugge la sessione attiva corrente se esiste.

View Solamente `_form.html.erb`, `index.html.erb`, `new.html.erb` come implementazione standard.

3.8.3 Autorizzazione

L'autorizzazione delle risorse standard mappa i sette metodi di Resource ai permessi attivabili a una classe di utenti come nella tabella *Mappa permessi-metodi*.

Tabella 2: Mappa permessi-metodi

Metodi	Permesso
new, show, index	read
create	create
update, edit	update
destroy	destroy

Per l'autorizzazione è prima necessario un sistema di autenticazione, che è implementato tramite la classe *Session*. E' stata utilizzata una libreria esterna per il caricamento filtrato e l'autorizzazione delle risorse. Il funzionamento della libreria è conosciuto dai membri del gruppo assegnati. Nel momento in cui si controlla l'autorizzazione ad una risorsa viene istanziato un oggetto di tipo *Ability*.

Ability

Funzione della classe Garantisce ad un utente i permessi che gli spettano. I permessi sono stati descritti in maniera gerarchica.

Attributi

+ user

l'utente per il quale si vogliono controllare i permessi

Metodi

+ initialize(user)

Costruisce l'oggetto ability salvando l'utente nell'attributo user. Invoca il metodo `roles` sullo user e per ogni ruolo ricevuto lancia su se stesso il metodo corrispondente che garantisce all'utente i permessi corretti

+ guests

Definisce i permessi per gli utenti non registrati.

+ desktop_user

Definisce i permessi degli utenti desktop.

+ mobile_users

Definisce i permessi degli utenti mobile.

+ superusers

Definisce i permessi dei superuser.

+ admins

Definisce i permessi degli admin

3.8.4 PagesController

Il controller PagesController, descritto nel file `app/controllers/pages_controller.rb`, gestisce le pagine statiche di Woty. Deriva da ApplicationController, non ha modello associato. Ad ogni metodo corrisponde una specifica vista contenente la pagina statica richiesta.

+ home

Richiamato dalla richiesta della pagina `/home`, visualizza l'homepage del sito, descritta nella vista contenuta nel file `/app/view/pages/home.html.erb`

+ pricing

Richiamato dalla richiesta della pagina `/pricing`, visualizza la lista dei Plan resi pubblici, come descritto nella vista contenuta nel file `/app/view/pages/pricing.html.erb`

+ contacts

Richiamato dalla richiesta della pagina `/contacts`, visualizza le informazioni di contatto del team SevenFold, come descritto nella vista contenuta nel file `/app/view/pages/contacts.html.erb`

+ denied

Richiamato dalla richiesta della pagina `/denied`, usualmente in seguito ad una richiesta non permessa all'utente, visualizza un avviso all'utente, come descritto nella vista contenuta nel file `/app/view/pages/denied.html.erb`

+ about

Richiamato dalla richiesta della pagina `/about`, visualizza delle informazioni su Woty, come descritto nella vista contenuta nel file `/app/view/pages/about.html.erb`

3.8.5 Moduli

Achievable::User Ogni metodo del modulo deve descrivere un metodo che ritorna un valore che può essere usato per costruire una condition.

+ `number_of_comments`

Ritorna il numero di commenti effettuati da un utente.

+ `number_of_status_changes`

Ritorna il numero di cambi di stato effettuati da un utente.

Requirements(incompleto) Ogni metodo del modulo rappresenta un requisito di sistema che deve essere validato.

+ `self.there_should_be_an_admin`

True se esiste almeno un admin.

+ `self.there_should_be_a_customer`

True se esiste almeno un customer,

+ `self.there_should_be_a_plan`

True se esiste almeno un plan.

Stats::User(incompleto) Ogni metodo definisce dei valori notevoli a fini statistici per gli user.

+ `number_of_recent_streams`

Numero di stream recenti visualizzabili dallo user

+ `social_points`

Punteggio relativo ad azioni sociali

4 Specifica Woty Desktop

Verrà di seguito presentata l'architettura in dettaglio dell'applicazione Woty Desktop per la ricezione delle notifiche di uno Desktop-user, attraverso una presentazione top-down.

Si inizierà analizzando l'interazione tra i componenti dell'applicazione, si proseguirà descrivendo i singoli componenti e l'interazione tra le classi, ed in fine verrà illustrata ogni singola classe indicandone attributi e metodi con rispettiva descrizione.

4.1 Diagramma Componenti

Di seguito è presentato il diagramma UML dei componenti riguardante l'applicazione Woty Desktop. Per una trattazione più completa e dettagliata dei componenti e delle classi si rimanda alla sezione 4.2.

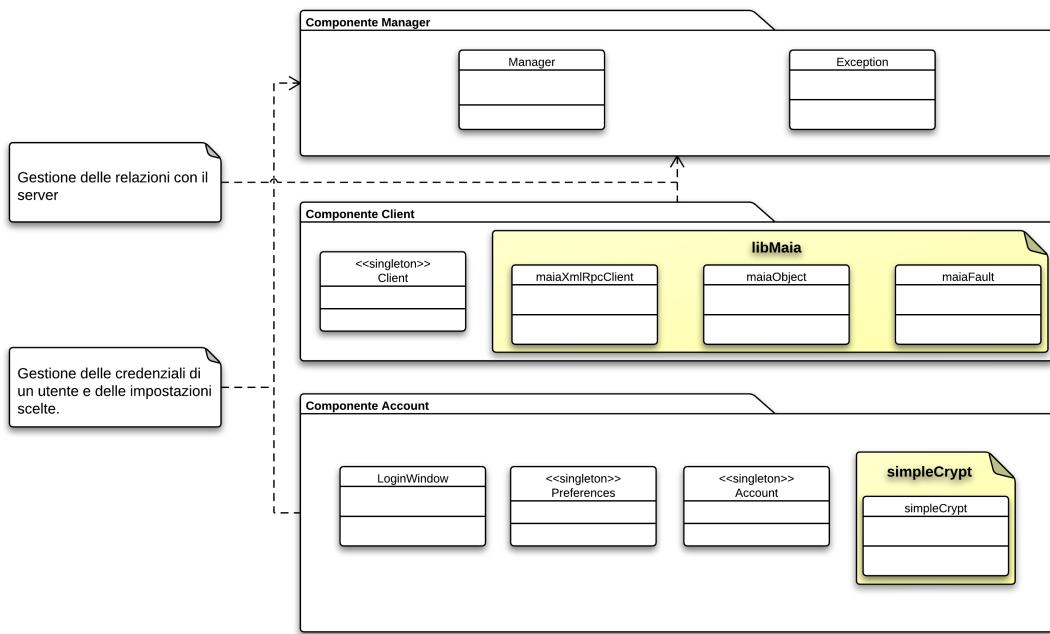


Figura 1: Desktop - Diagramma Componenti

4.2 Diagramma Classi

Di seguito è presentato il diagramma UML dei delle classi riguardante l'applicazione Woty Desktop.

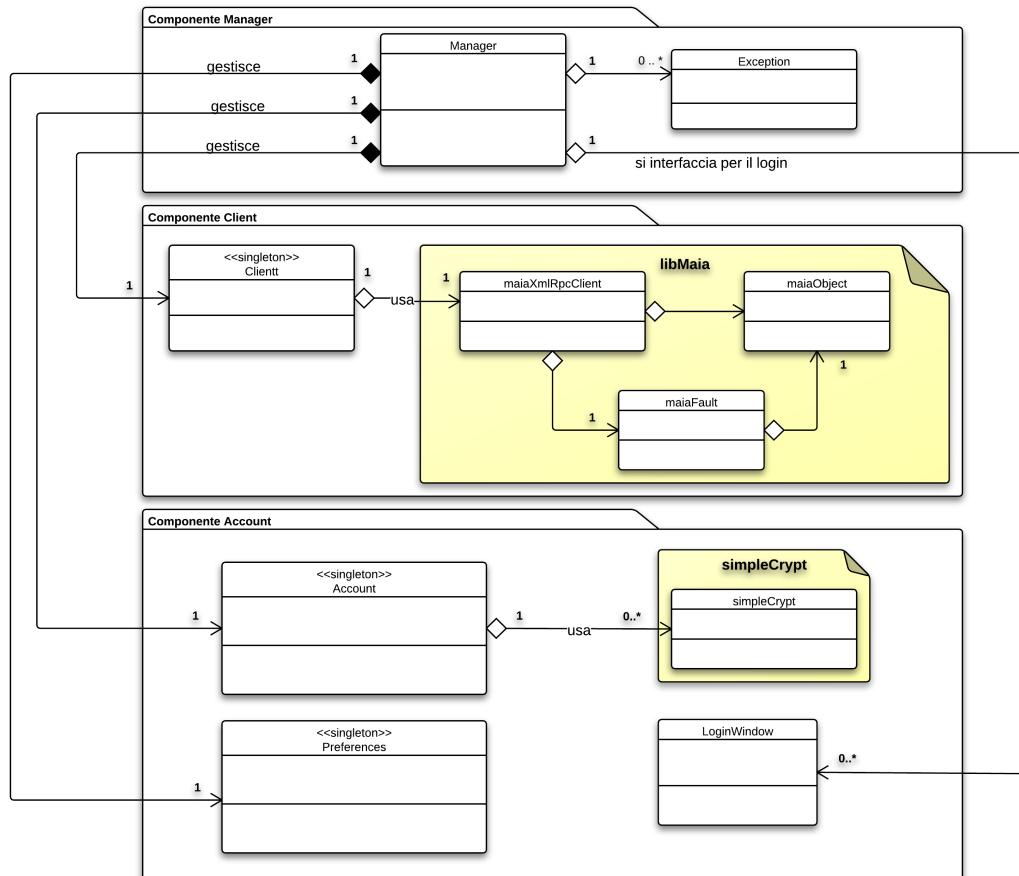


Figura 2: Desktop - Diagramma Classi

4.3 Specifica componente Manager

Il componente Manager ha il compito di gestire l'applicazione Woty Desktop, interagendo con gli altri componenti del sottosistema e coordinandoli tra di loro.

4.3.1 Specifica classe Manager

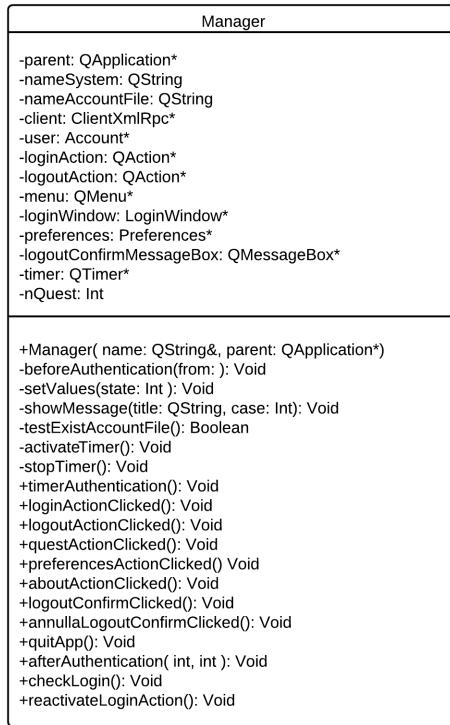


Figura 3: Desktop - Classe Manager

Funzione della classe

La classe Manager consiste nel "core" dell'applicazione Woty Desktop, avvia ogni funzionalità quali il login, le comunicazioni con la piattaforma Woty per l'invio delle credenziali e la ricezione di notifiche e il controllo delle preferenze dell'utente.

Relazioni con altre classi

- Utilizza le classi:
 - **Exception**
 - **Client**
 - **LoginWindow**
 - **Account**
 - **Preferences**

Attributi

- **QApplication* parent**
Puntatore all'applicazione Qt.
- **QString nameSystem**
Attributo che memorizza il nome del sistema.
- **QString nameAccountFile**
Attributo che memorizza il percorso e nome del file account.dat in cui vengono salvati i dati dell'utente.

- Client* client

Puntatore all'istanza statica della classe Client che gestisce le comunicazioni con la piattaforma Woty.

- Account* user

Puntatore all'istanza statica della classe Account per la gestione dell'utente utilizzatore.

- QAction* loginAction

Puntatore all'azione di Login del menù.

- QAction* logoutAction

Puntatore all'azione di Logout del menù.

- QMenu* menu

Puntatore al menù grafico dell'applicazione.

- LoginWindow* loginWindow

Puntatore all'istanza statica della classe LoginWindow, per l'inserimento delle credenziali di un utente.

- Preferences* preferences

Puntatore all'istanza statica della classe Preferences per la gestione delle preferenze dell'utente.

- QMessageBox* logoutConfirmMessageBox

Puntatore alla finestra per la conferma del logout dell'utente.

- QTimer* timer

Puntatore al timer che temporizza l'invio delle richieste al server per il controllo di nuove notifiche.

- int nQuest

Attributo che memorizza in numero di nuove notifiche dell'utente loggato.

Metodi

+ Manager(const QString& name, QApplication* parent)

Costruttore delle classe Manager che riceve in input il nome dell'applicazione, e il puntatore alla qApp.

Crea il menù con le opzioni e le rispettive connect per la gestione dei signal e degli slot.

- void beforeAuthentication(int from)

All'avvio dell'applicazione viene invocato dal costruttore della classe Manager e controlla se l'utente aveva scelto di memorizzare le proprie credenziali di accesso alla piattaforma Woty, e in caso affermativo avvia il client per eseguire il login.

Nel caso in cui invece l'utente non sia loggato e decida di eseguire il login, il metodo recupera le credenziali di accesso dalla form di login, e avvia il client per eseguire il login. Il parametro from, tiene traccia se l'esecuzione è arrivata del costruttore della classe Manager oppure dalla form per il login "loginWindow".

- void setValues(int x)

Metodo che setta correttamente l'icona della Tray Icon⁹ in base a questi quattro possibili stati:

- l'utente è loggato alla piattaforma Woty e non ha notifiche in sospeso,
- l'utente è loggato alla piattaforma Woty e ha notifiche in sospeso,
- l'utente è sloggato dalla piattaforma Woty,
- l'utente ha memorizzato le proprie credenziali di accesso alla piattaforma Woty, ma non è connesso alla rete.

- void showMessage(QString title, int x)

Metodo che permette di visualizza messaggi di notifiche per l'utente.

Il metodo consiste in un override del metodo QSystemTrayIcon::showMessage().

- bool testExistAccountFile()

Metodo che testa l'esistenza del file account/account.dat che contiene le credenziali di autenticazione alla piattaforma Woty dell'utente.

- void activateTimer()

Metodo che attiva il timer per il controllo periodico temporizzato della presenza di nuove notifiche, e che imposta il valore temporale degli intervalli di tempo.

Viene invocato quando viene eseguito correttamente un login alla piattaforma Woty.

- void stopTimer()

Metodo che ferma il timer per il controllo periodico temporizzato della presenza di nuove notifiche.

Viene invocato al logout di un utente.

+ void timerAuthentication()

Metodo slot che viene richiamato automaticamente ad ogni timeout del timer.

Il metodo invoca il client per effettuare il controllo alla piattaforma Woty della presenza di nuove notifiche.

+ void loginActionClicked()

Metodo slot che viene invocato al premere dell'utente dell'opzione "Login" del menù. Se le credenziali dell'utente erano state precedentemente memorizzate, esegue il login automaticamente invocando il client, altrimenti crea la form per l'inserimento delle credenziali.

+ void logoutActionClicked()

Metodo slot che viene invocato al premere dell'utente dell'opzione "Logout" del menù. Crea un messaggio che richiede all'utente la conferma del logout.

+ void questActionClicked()

Metodo slot che viene invocato al premere dell'utente dell'opzione "Svolgi Quest" del menù. Il metodo apre il browser predefinito del sistema operativo e se l'utente è loggato all'applicazione Woty Web accede direttamente alla pagina relativa alle quest, altrimenti richiede il login.

+ void preferencesActionClicked()

Metodo slot che viene invocato al premere dell'utente dell'opzione "Preferenze" del menù. Il metodo visualizza la finestra per la scelta delle preferenze.

+ void aboutActionClicked()

Metodo slot che viene invocato al premere dell'utente dell'opzione "Informazioni" del menù. Il metodo apre il browser predefinito del sistema operativo e accede alla pagina *About* dell'applicazione Woty Web.

+ void logoutConfirmClicked()

Metodo slot che viene invocato alla conferma dell'utente di voler effettuare il logout. Il metodo esegue il logout cancellando le informazioni riguardanti le credenziali di accesso dell'utente eliminando i campi `mail` e `psw` dell'istanza statica della classe `Account`, ed eliminando il file `account/account.dat` nel caso in cui l'utente avesse scelto di memorizzare le proprie credenziali.

Il metodo inoltre cambia l'icona dell'applicazione impostando l'icona riguardante lo stato di logout.

+ void annullaLogoutConfirmClicked()

Metodo slot che distrugge il messaggio di conferma logout.

+ void quitApp()

Metodo slot che chiude l'applicazione e dealloca i campi dati.

+ void afterAuthentication(int from, int result)

Metodo slot che viene processa le risposte del server ricevute dal client ad ogni tentativo di login, o di ricerca di nuove notifiche.

Il campo result può contenere vari valori, secondo queste specifiche:

- result == -1 indica che l'autenticazione al server è fallita,
- result >-1 indica il numero di notifiche riguardanti l'utente: l'autenticazione è andata a buon fine,
- result == -100 indica che il terminale non è connesso alla rete.

Il campo from indica se la richiesta al server è stata richiamata del costruttore della classe Manager durante l'avvio dell'applicazione o successivamente per comando dell'utente o per timeout della temporizzazione del timer.

+ void checkLogin()

Metodo slot che richiama il metodo beforeAuthentication(int) in seguito al completamento della form per il login di un utente.

+ void reactivateLoginAction()

Metodo slot che riattiva l'opzione di "Login" dal menù quando un utente esegue il logout.

4.3.2 Specifica classe Exception

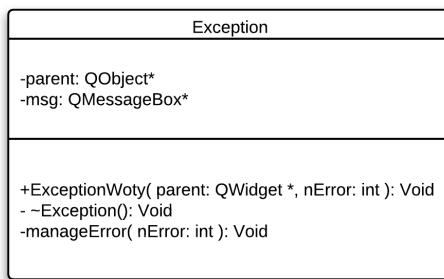


Figura 4: Desktop - Classe Exception

Funzione della classe

La classe Exception ha il compito di gestire le eccezioni e gli errori gestiti che possono essere sollevati durante l'esecuzione dell'applicazione. Ogni eccezione o errore è identificato da un numero, in base al quale la classe processa in modo specifico l'eccezione o errore gestendolo e/o lanciando un messaggio di errore.

Relazioni con altre classi

- E' utilizzata dalla classe:

- Manager

Attributi

+ QWidget* parent

Puntatore alla classe gestore Manager.

- `QMessageBox* msg`

Puntatore al messaggio di errore lanciato visualizzato.

Metodi

+ `Exception(QWidget *parent, int nError)`

Costruttore della classe Exception che riceve in input il puntatore alla classe gestore Manager e il numero dell'errore sollevato.

- `void manageError(int nError)`

Metodo che riceve in input il numero dell'errore da processare.

4.4 Specifica componente Client

Il componente Client ha il compito di gestire l'autenticazione di un utente alla piattaforma Woty attraverso l'invio al server delle credenziali di accesso e la ricezione delle risposte del server. Inoltre ad autenticazione avvenuta invia periodicamente richieste al server per il controllo della presenza di nuove notifiche.

All'interno del componente viene utilizzata la libreria libMaia per l'implementazione del client xml rpc.

4.4.1 Specifica classe Client

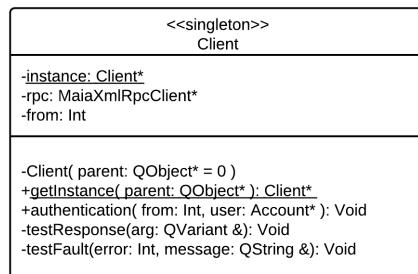


Figura 5: Desktop - Classe Client

Funzione della classe

La classe Client invia richieste al server Woty: richiede l'autenticazione per il login e invia richieste per la presenza di notifiche, ricevendo le risposte del server.

Relazioni con altre classi

- Utilizza le classi:

- `maiaXmlRpcClient`

- È utilizzata dalle classi:

- `Manager`

Attributi

- `static Client* instance`

Puntatore all'unica istanza della classe singleton Client.

- `MaiaXmlRpcClient *rpc`

Istanza della classe MaiaXmlRpcClient della libreria libMaia.

- `int from`

Campo dati che tiene traccia da dove è stata invocata l'ultima richiesta del client.

Metodi

- `Client(QObject* parent = 0)`

Costruttore privato della classe Client; riceve il puntatore al padre e alloca gli oggetti MaiaXmlRpcClient* rpc. Setta inoltre l'URL del server Woty.

+ `static Client* getInstance(QObject* parent)`

Metodo statico che restituisce l'unica istanza della classe Client secondo il design pattern Singleton.

+ `void authentication(int from, Account* user)`

Metodo che invoca l'autenticazione di un utente sul server Woty. Riceve in campo from che indica da dove è arrivata la richiesta di autenticazione e un oggetto user contenete le informazioni riguardanti l'utente.

+ `void testResponse(QVariant& arg)`

Metodo che riceve le risposte del server ed invoca Manager::afterAuthenticated(int, int) che processerà la risposta del server.

+ `void testFault(int error, QString& message)`

Metodo che viene invocato se la connessione non è disponibile e il client restituisce un messaggio di errore.

4.4.2 Specifica libreria LibMaia

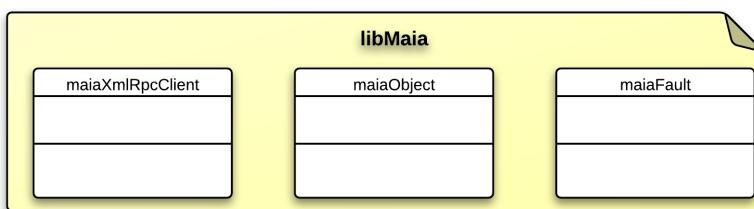


Figura 6: Desktop - libreria libMaia

Funzione della libreria

La libreria fornisce una struttura completa per l'implementazione di un client xml rpc.

Relazioni con altre classi

- La libreria è utilizzata dalle classi:

- `Client`

4.5 Specifica componente Account

Il componente Account ha il compito di gestire l'inserimento delle credenziali di accesso di un utente attraverso una semplice form per il login.

Inoltre si occupa del salvataggio di tali credenziali quali e-mail e password, e della criptazione della password qualora l'utente decida di rimanere loggato al sistema.

In tale componente viene utilizzata la libreria SimpleCrypt per la criptazione.

4.5.1 Specifica classe LoginWindow

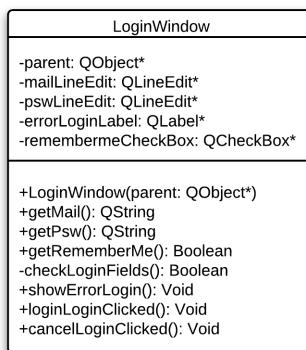


Figura 7: Desktop - Classe LoginWindow

Funzione della classe

La classe LoginWindow consiste in una finestra grafica contenente una form per l'inserimento delle credenziali di accesso di un utente.

Relazioni con altre classi

- La libreria è utilizzata dalle classi:

- Manager

Attributi

- `QObject* parent`

Puntatore al padre Manager.

- `QLineEdit* mailLineEdit`

Casella di testo per l'inserimento della mail.

- `QLineEdit* pswLineEdit`

Casella di testo per l'inserimento della password che non verrà visualizzata in chiaro.

- `QLabel* errorLoginLabel`

Puntatore ad una label che indica un messaggio di insuccesso di un login, e quindi che i dati inseriti non sono corretti.

- `QCheckBox* remembermeCheckBox`

Check box per permettere all'utente di selezionare il salvataggio delle proprie credenziali.

Metodi

+ `LoginWindow(QObject* parent)`

Costruttore LoginWindow che crea la finestra contenente la form per il login.

+ `QString getMail()`

Metodo che restituisce la mail inserita dall'utente.

+ `QString getPsw()`

Metodo che restituisce la password inserita dall'utente.

+ `bool getRememberme()`

Metodo che restituisce un booleano che indica se l'utente vuole memorizzare le proprie credenziali di accesso.

- `bool checkLoginFields()`

Metodo che fa una prima verifica sul campo mail inserito dall'utente, controllando il corretto formato della mail.

+ `void showErrorLogin()`

Metodo che visualizza nella form di login la QLabel* errorLoginLabel che indica che l'autenticazione al server non è andata a buon fine: le credenziali di accesso inserite dall'utente non sono corrette.

+ `void loginLoginClicked()`

Metodo che richiama checkLoginFields() per il controllo sul corretto formato della mail, e successivamente se il controllo è andato a buon fine, avvisa il Manager della richiesta dell'utente di effettuare il login.

+ `void cancelLoginClicked()`

Metodo che distrugge l'istanza dell'oggetto LoginWindow chiudendo la form di login.

4.5.2 Specifica classe Account

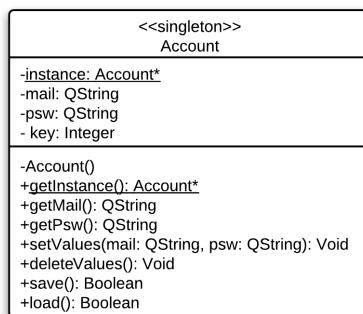


Figura 8: Desktop - Classe Account

Funzione della classe

Classe singleton Account ha il compito di gestire le credenziali di un utente.

Relazioni con altre classi

- Utilizza le classi:
 - simpleCrypt (libreria)
- È utilizzata dalle classi:
 - Manager

Attributi

- **static Account* instance**
Puntatore all'unica istanza della classe singleton Account.
- **QString mail**
Attributo che memorizza la mail inserita dall'utente.
- **QString psw**
Attributo che memorizza la password inserita dall'utente.
- **quint64 key**
Attributo che memorizza una chiave che sarà utilizzata per criptare la password.

Metodi

- **Account()**
Costruttore privato per la classe Account.
- + **static Account* getInstance()**
Metodo statico che ritorna l'unica istanza della classe Account.
- + **const QString& getMail()**
Metodo che ritorna la password dell'utente.
- + **const QString& getPsw()**
Metodo che ritorna la mail dell'utente.
- + **void setValues(const QString& mail, const QString& psw)**
Metodo che inserisce le credenziali dell'utente nei rispettivi campi mail e psw.
- + **void deleteValues()**
Metodo che elimina la memorizzazione delle credenziali dell'utente quando viene effettuato un logout.
- + **bool save()**
Metodo che viene invocato quando un utente effettua un login scegliendo di rimanere loggato.
Il metodo controlla l'esistenza della cartella *account* con al suo interno il file *account.dat* e in caso non esistano li crea. Successivamente cripta la password utilizzando la libreria simpleCrypt ed in fine salva nel file *account/account.dat* le credenziali.
- + **bool load()**
Metodo che carica le credenziali dell'utente leggendole dal file *account/account.dat*. Successivamente il metodo deccripta la password attraverso l'uso della libreria simpleCrypt e memorizza i dati nei campi dati mail e psw dell'unica istanza della classe Account.

4.5.3 Specifica libreria simpleCrypt

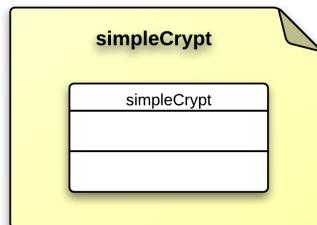


Figura 9: Desktop - libreria SimpleCrypt

Funzione della libreria

La libreria simpleCrypt ha la funzione di criptare o decriptare delle stringhe attraverso l'utilizzo di una chiave: ricevere in input la stringa da criptare o decriptare e la chiave, restituendone la stringa criptata.

Viene utilizzata dalla classe Account per il criptazione della password utente.

Relazioni con altre classi

- È utilizzata dalle classi:

- Account

4.5.4 Specifica classe Preferences

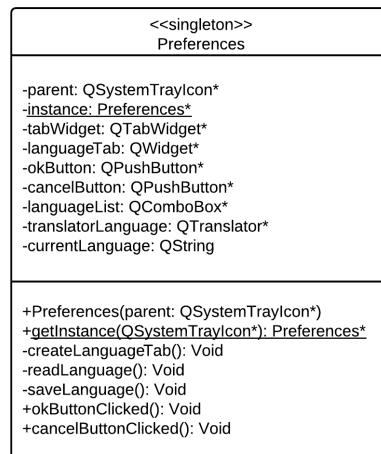


Figura 10: Desktop - Classe Preferences

Funzione della classe

La classe consiste in una finestra grafica attraverso la quale l'utente può impostare le proprie preferenze dell'applicazione.

Relazioni con altre classi

- È utilizzata dalle classi:
 - Manager

Attributi

- **QSystemTrayIcon* parent**
Puntatore all'istanza della classe Manager.
- **static Preferences* instance**
Puntatore all'unica istanza della classe singleton Preferences.
- **QTabWidget* tabWidget**
Attributo grafico per la creazione della finestra.
- **QWidget* languageTab**
Attributo grafico che consiste nel tab riguardante l'impostazione della lingua.
- **QPushButton* okButton**
Puntatore al bottone di conferma delle modifiche delle impostazioni.
- **QPushButton* cancelButton**
Puntatore al bottone di cancellazione delle modifiche delle impostazioni.
- **QComboBox* languageList**
Puntatore al combo box per la selezione della lingua.
- **QTranslator* translatorLanguage**
Puntatore all'oggetto `translator` per la gestione delle lingue.
- **QString currentLanguage**
Campo dati che memorizza la lingua corrente impostata.

Metodi

- **Preferences(QSystemTrayIcon* parent)**
Costruttore della classe Preferences che crea la finestra grafica.
- + **static Preferences* getInstance()**
Metodo che restituisce l'unica istanza della classe Preferences.
- **void createLanguageTab()**
Metodo che crea il tab per l'impostazione della lingua. Il metodo controlla dinamicamente nella cartella *languages* quali lingue di traduzione sono disponibili e per ogni rispettiva lingua trovata crea l'opzione di selezione inserendola nell'attributo `QComboBox* languageList`.
- **void readLanguage()**
Metodo che legge dal file *languages/language.dat* la lingua impostata dall'utente.
- **void saveLanguage()**
Metodo che salva la lingua impostata dall'utente nel file *languages/languages.dat* ad ogni cambiamento della lingua dell'applicazione eseguito dall'utente.

+ void **okButtonClicked()**

Metodo che viene invocato al click del pulsante QPushButton* okButton. Il metodo controlla se l'utente ha effettuato dei cambiamenti sulle impostazioni dell'applicazione. In caso affermativo il metodo rende effettivo il cambiamento e nasconde la finestra delle preferenze.

+ void **cancelButtonClicked()**

Il metodo viene invocato al click del pulsante QPushButton* cancelButton, e serve per effettuare il ripristino del settaggio delle impostazioni nella finestra delle preferenze. Il metodo inoltre nasconde la finestra delle preferenze.

5 Specifica Woty Mobile

Woty Mobile consiste nell'applicazione dedicata ai Mobile-User utilizzatori di un dispositivo Android. L'applicativo dovrà fornire tutte le funzionalità dedicate ai Desktop-User compreso il sistema di notifica dell'arrivo di nuove quest e la risoluzione dal dispositivo delle stesse. Dopo che l'utente avrà effettuato il login alla piattaforma Woty, avrà accesso a tutte le funzionalità del sistema.

5.1 Diagramma Componenti

Di seguito è presentato il diagramma UML dei componenti riguardante l'applicazione Woty Mobile. Per una trattazione più completa e dettagliata dei componenti e delle classi si rimanda alle sezioni successive.

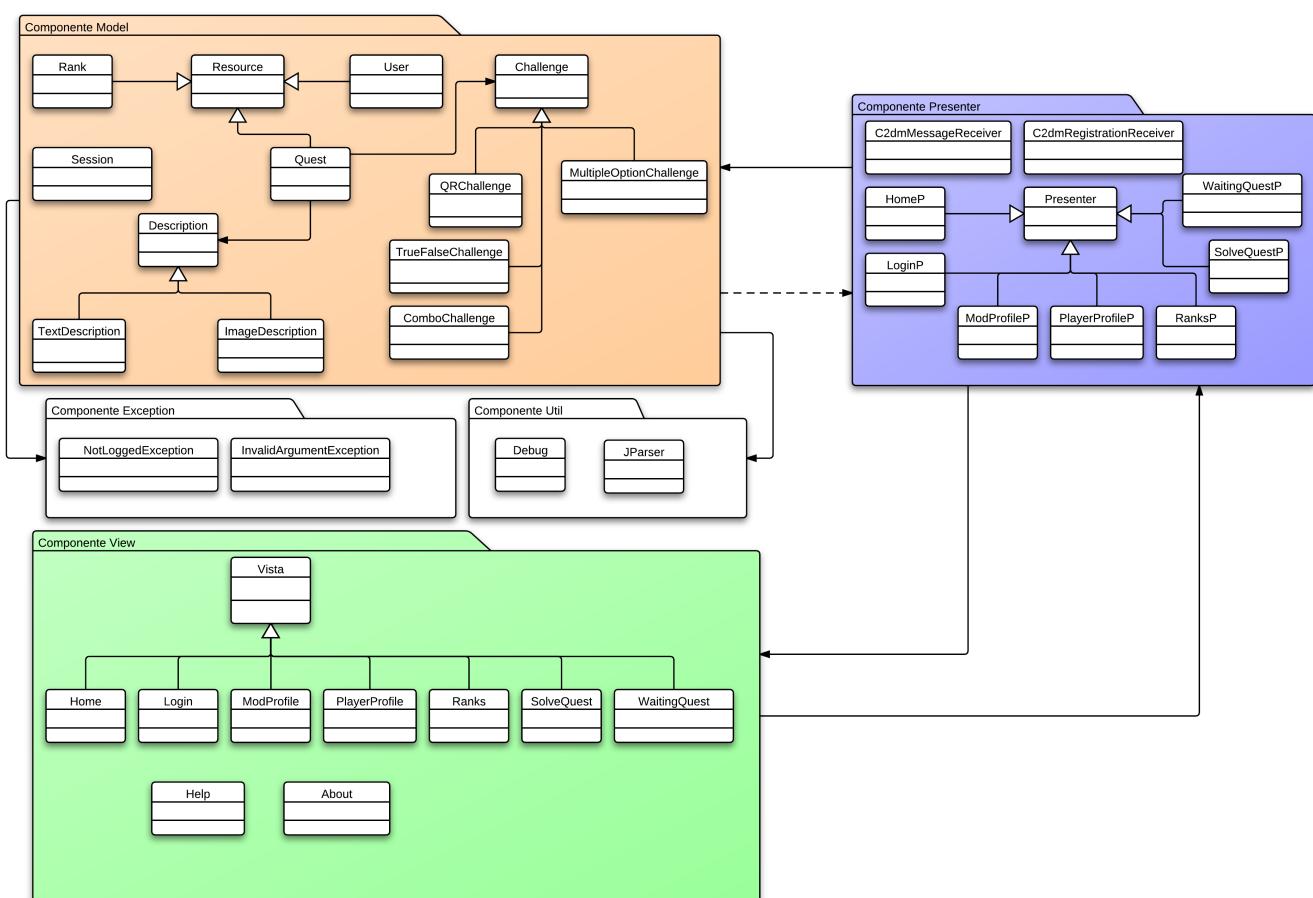


Figura 11: Mobile - Diagramma Componenti

5.2 Diagramma delle Classi

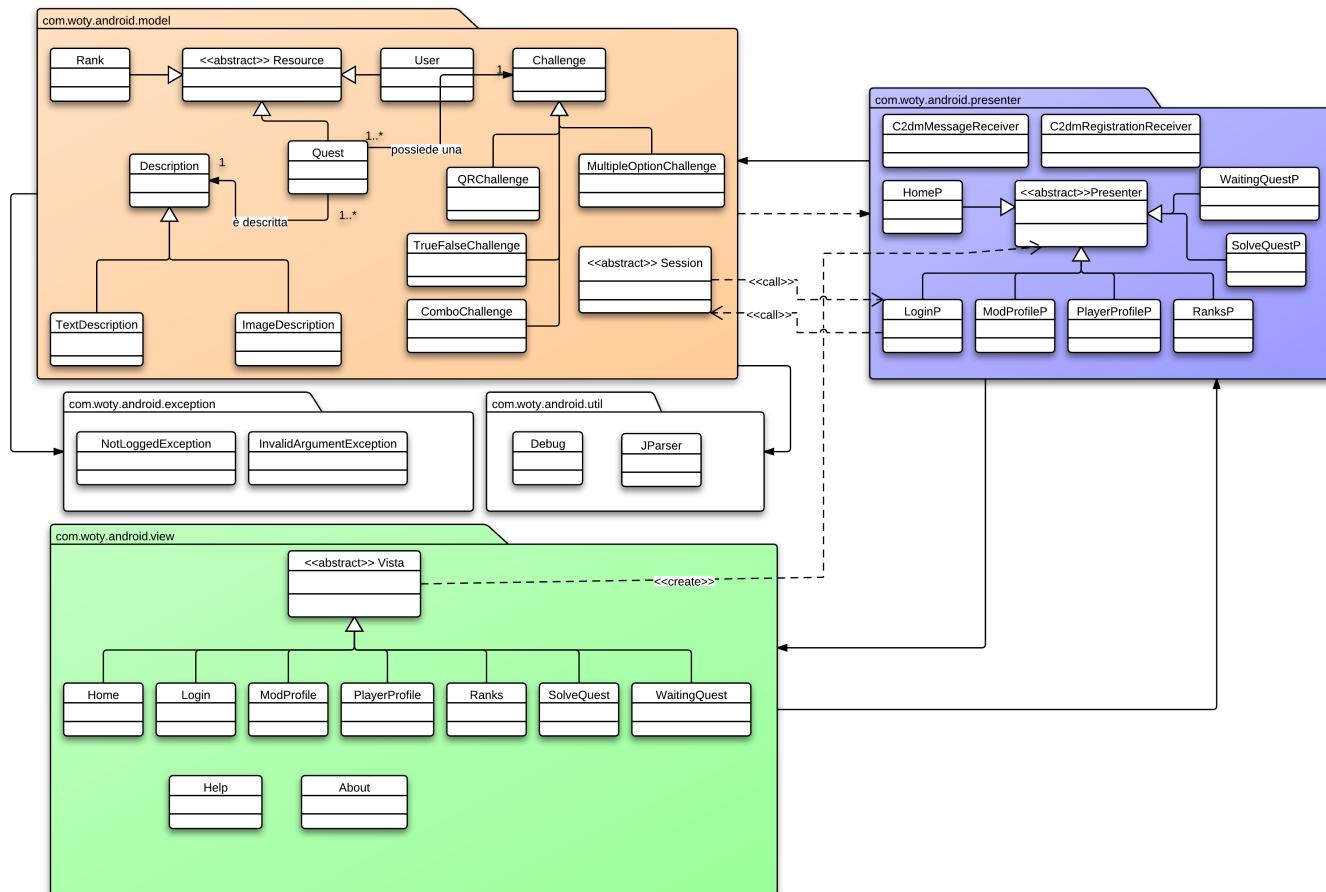


Figura 12: Mobile - Diagramma Classi

5.3 Specifica componente Model

Componente che contiene le classi rappresentanti le entità logiche dell'applicazione e corrisponde per l'appunto al componente Model del pattern MVP.

5.3.1 com.woty.android.model.Resource

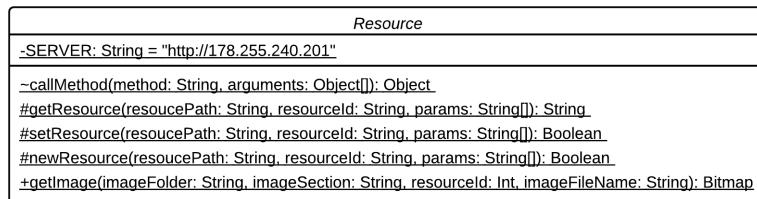


Figura 13: Classe com.woty.android.model.Resource

Funzione della classe

Classe astratta che rappresenta una generica risorsa del modello dai dati, definendo alcuni attributi e metodi comuni. Sarà estesa da tutte le classi del componente. Fornisce inoltre i metodi generici per effettuare richieste al server Woty. Per la definizione di questi metodi ci si appoggia alle librerie org.xmlrpc.android e org.apache.http.

Relazioni con altre classi

- È estesa dalle classi:
 - com.woty.android.model.User
 - com.woty.android.model.Challenge
 - com.woty.android.model.Description
 - com.woty.android.model.Quest
 - com.woty.android.model.QuestHistory
 - com.woty.android.model.Rank
- Utilizza le classi:
 - org.xmlrpc.android.XMLRPCClient
 - org.apache.http.client.HttpClient

Attributi

```
- private static final String SERVER = "http://178.255.240.201"
```

Stringa contenente l'indirizzo del server.

Metodi

- ~ static final Object callMethod(String method, Object[] arguments)

Metodo utilizzato per invocare un generico metodo xml-rpc.
- # static final String getResource(String resourcePath, String resourceId, String[] params)

Metodo che ottiene una generica risorsa dal server fornendo i relativi identificatori.
- # static final boolean setResource(String resourcePath, String resourceId, String[] params)

Metodo che modifica una generica risorsa sul server fornendo i relativi identificatori gli attributi da modificare (params).

```
# static final boolean newResource(String resoucePath, String resourceId, String[]
    params)
    Metodo che crea una nuova risorsa generica sul server fornendo i relativi dati.

+ static final Bitmap getImage(String imageFolder, String imageSection, int resourceId,
    String imageFileName)
    Metodo che ottiene un'immagine generica dal server.
```

5.3.2 com.woty.android.model.User

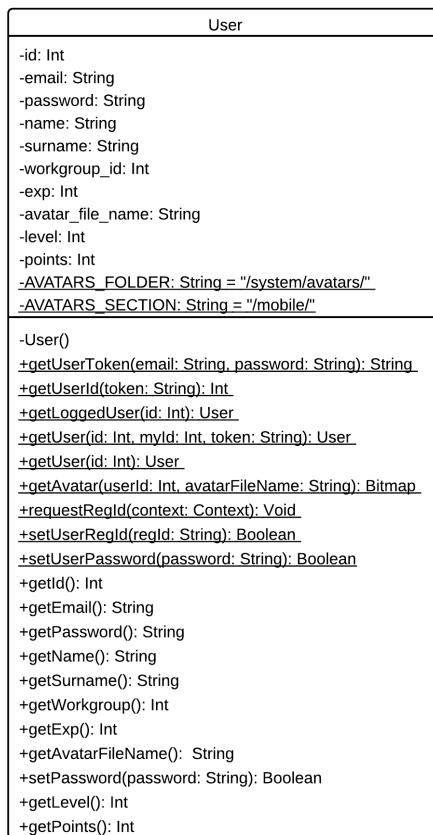


Figura 14: Classe com.woty.android.model.User

Funzione della classe

Classe che rappresenta un utente del sistema e ne raccoglie le informazioni.

Relazioni con altre classi

- Estende la classe com.woty.android.model.Resource
- È utilizzata da:
 - com.woty.android.model.Session
- Utilizza le classi:
 - com.woty.android.util.JParser
 - com.woty.android.exceptions.NotLoggedException

Attributi

- `int id`
Numero identificativo univoco dell'utente.
- `String email`
Indirizzo email dell'utente.
- `String password`
Password di accesso al sistema dell'utente.
- `String name`
Nome dell'utente.
- `String surname`
Cognome dell'utente.
- `int workgroup_id`
Numero identificativo del workgroup a cui l'utente è associato.
- `int exp`
Punteggio totale dell'utente.
- `String avatar_file_name`
Nome dell'immagine usata dall'utente come avatar.
- `int points`
Indica il punteggio totale ottenuto dall'utente date le quest svolte.
- `int level`
Indica il livello dell'utente.
- `static final String AVATARS_FOLDER = "/system/avatars/"`
Stringa contenente l'indirizzo della cartella degli avatar da appendere all'indirizzo del server.
- `static final String AVATARS_SECTION = /mobile/`
Stringa contenente l'indirizzo della sezione "mobile" degli avatar.

Metodi

- `User()`
Viene definito il costruttore privato in quanto uno User è istanziabile solamente in ritorno da una richiesta al server.
- + `static final String getUserToken(String email, String password)`
Metodo che ritorna il token assegnato all'utente autenticato dati email e password dello stesso.
- + `static final int getUserId(String token)`
Metodo che ritorna il numero identificativo dell'utente autenticato.
- + `static final User getLoggedUser(int id)`
Metodo che ritorna il riferimento all'utente autenticato nella sessione corrente.
- + `static final User getUser(int id, int myId, String token)`
Metodo che ritorna il riferimento ad un utente passando relativi id e token. Questo metodo è utilizzato al login di un utente.
- + `static final User getUser(int id) throws NotLoggedException`
Metodo che ritorna il riferimento all'utente autenticato passando relativi id e token e solleva un'eccezione di tipo `NotLoggedException` nel caso in cui la sessione corrente non sia autenticata.

```
+ static final Bitmap getAvatar(int id, String avatarFileName)
    Metodo che ottiene un avatar dal server fornendo identificativo dello User possessore e
    il nome dell'immagine.

+ static void requestRegId(Context context)
    Metodo che effettua la richiesta di una nuovo registration id per il servizio C2DM dello
    User.

+ static final boolean setUserRegId(String regId) throws NotLoggedException
    Metodo che modifica il registration id nel server per il servizio C2DM dello User.

+ static final boolean setPassword(String password) throws NotLoggedException
    Metodo che modifica la password dello User passando identificativo, token dello stesso e
    la nuova password.
```

Per tutti gli attributi della classe devono essere implementati i metodi *get*.
 I metodi *set* da implementare saranno solo quelli dei campi modificabili dall'utente:

```
+ void setPassword(String p)
    Metodo che modifica la password dell'utente.

+ void setAvatar(String s)
    Metodo che modifica l'avatar dell'utente.
```

5.3.3 com.woty.android.model.Challenge

<i>Challenge</i>
<pre>#id: Int #name: String #question: String #tfstatements: HashMap<String, String> #Challenge() +getChallenge(challengerId: Int, type: String): Challenge +toStrings(): String[] +getId(): Int +getName(): String +getQuestion(): String +getRtfstatements(): HashMap<String, String></pre>

Figura 15: Classe com.woty.android.model.Challenge

Funzione della classe

Classe astratta che rappresenta una sfida generica che sarà offerta in una quest del sistema e ne raccoglie le informazioni.

Relazioni con altre classi

- Estende la classe `com.woty.android.model.Resource`
- È estesa dalle classi:
 - `com.woty.android.model.TrueFalseChallenge`
 - `com.woty.android.model.ComboChallenge`
 - `com.woty.android.model.MultiOptionChallenge`
 - `com.woty.android.model.QRChallenge`
- Utilizza le classi:
 - `com.woty.android.util.JParser`
 - `com.woty.android.exceptions.NotLoggedException`

Attributi

```

# int id
    Identificativo della sfida.

# String name
    Nome della sfida.

# String question
    Domanda presente nella sfida.

# HashMap<String, String> tfstatements
    HashMap che contiene tutte le possibilità di risposta alla sfida date dal server.

```

Metodi

```

# Challenge()
    Costruttore protetto della classe.

+ static Challenge getChallenge(int challengeId, String type) throws NotLoggedException
    Metodo che ritorna l'istanza dell'oggetto Challenge dal server dato il suo identificativo
    e il tipo.

+ String[] toStrings()
    Metodo che ritorna una stringa rappresentante l'oggetto.

+ int getId()
    Metodo getter dell'attributo id.

+ String getName()
    Metodo getter dell'attributo name.

+ String getQuestion()
    Metodo getter dell'attributo question.

+ HashMap<String, String> getRtfstatements()
    Metodo getter dell'attributo rtfStatements.

```

5.3.4 com.woty.android.model.TrueFalseChallenge

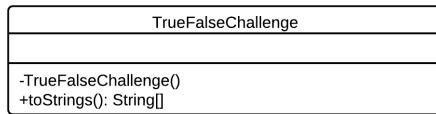


Figura 16: Classe com.woty.android.model.TrueFalseChallenge

Funzione della classe

Classe astratta che rappresenta una sfida con domanda vero o falso che sarà offerta in una quest del sistema e ne raccoglie le informazioni.

Relazioni con altre classi

- Estende la classe com.woty.android.model.Challenge

Attributi

Metodi

- `TrueFalseChallenge()`

Viene definito il costruttore privato in quanto una `TrueFalseChallenge` è istanziabile solamente in ritorno da una richiesta al server.

+ `String[] toStrings()`

Metodo ridefinito dalla superclasse che ritorna una stringa rappresentante l'oggetto.

5.3.5 com.woty.android.model.ComboChallenge

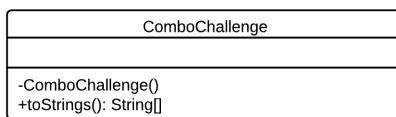


Figura 17: Classe com.woty.android.model.ComboChallenge

Funzione della classe

Classe astratta che rappresenta una domanda la cui risposta dovrà essere scelta tra più possibilità che sarà offerta in una quest del sistema e ne raccoglie le informazioni.

Relazioni con altre classi

- Estende la classe `com.woty.android.model.Challenge`

Attributi

Metodi

- `ComboChallenge()`

Viene definito il costruttore privato in quanto una `ComboChallenge` è istanziabile solamente in ritorno da una richiesta al server.

+ `String[] toStrings()`

Metodo ridefinito dalla superclasse che ritorna una stringa rappresentante l'oggetto.

5.3.6 com.woty.android.model.MultiOptionChallenge

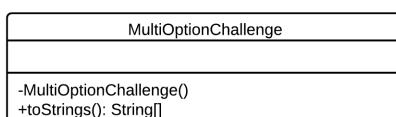


Figura 18: Classe com.woty.android.model.MultiOptionChallenge

Funzione della classe

Classe astratta che rappresenta una sfida con risposte multiple che sarà offerta in una quest del sistema e ne raccoglie le informazioni.

Relazioni con altre classi

- Estende la classe `com.woty.android.model.Challenge`

Attributi

Metodi

- MultiOptionChallenge()

Viene definito il costruttore privato in quanto una MultiOptionChallenge è istanziabile solamente in ritorno da una richiesta al server.

+ String[] toStrings()

Metodo ridefinito dalla superclasse che ritorna una stringa rappresentante l'oggetto.

5.3.7 com.woty.android.model.QRChallenge

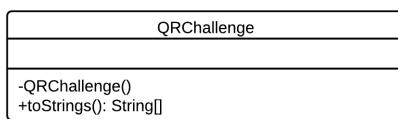


Figura 19: Classe com.woty.android.model.QRChallenge

Funzione della classe

Classe astratta che rappresenta una sfida con compito cognitivo che includa la scansione di QR-Code che sarà offerta in una quest del sistema e ne raccoglie le informazioni.

Relazioni con altre classi

- Estende la classe com.woty.android.model.Challenge

Attributi

Metodi

- QRChallenge()

Viene definito il costruttore privato in quanto una QRChallenge è istanziabile solamente in ritorno da una richiesta al server.

+ String[] toStrings()

Metodo ridefinito dalla superclasse che ritorna una stringa rappresentante l'oggetto.

5.3.8 com.woty.android.model.Description

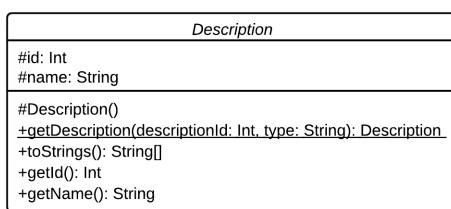


Figura 20: Classe com.woty.android.model.Description

Funzione della classe

Classe astratta che rappresenta una generica descrizione di una quest del sistema e ne raccoglie le informazioni.

Relazioni con altre classi

- Estende la classe `com.woty.android.model.Resource`
- È estesa dalle classi:
 - `com.woty.android.model.TextDescription`
 - `com.woty.android.model.ImageDescription`
- Utilizza le classi:
 - `com.woty.android.util.JParser`
 - `com.woty.android.exceptions.NotLoggedException`

Attributi

`int id`
Identificativo della descrizione.

`String name`
Nome della descrizione.

Metodi

`Description()`
Costruttore protetto della classe.

+ `static final Description getDescription(int descriptionId, String type) throws NotLoggedException`
Metodo che ritorna dal server l'istanza dell'oggetto `Description` passando identificativo e tipo della descrizione.

+ `String[] toStrings()`
Metodo che ritorna una stringa rappresentante l'oggetto.

+ `int getId()`
Metodo *getter* dell'attributo `id`.

+ `String getName()`
Metodo *getter* dell'attributo `name`.

5.3.9 com.woty.android.model.TextDescription

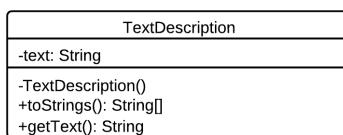


Figura 21: Classe com.woty.android.model.TextDescription

Funzione della classe

Classe che rappresenta una descrizione testuale di una quest del sistema e ne raccoglie le informazioni.

Relazioni con altre classi

- Estende la classe `com.woty.android.model.Description`

Attributi

- `private String text`

Stringa contenente il testo della descrizione.

Metodi

- `TextDescription()`

Viene definito il costruttore privato in quanto una `TextDescription` è istanziabile solamente in ritorno da una richiesta al server.

+ `String[] toStrings()`

Metodo ridefinito dalla superclasse che ritorna una stringa rappresentante l'oggetto.

+ `String getText()`

Metodo *getter* dell'attributo `text`.

5.3.10 com.woty.android.model.ImageDescription

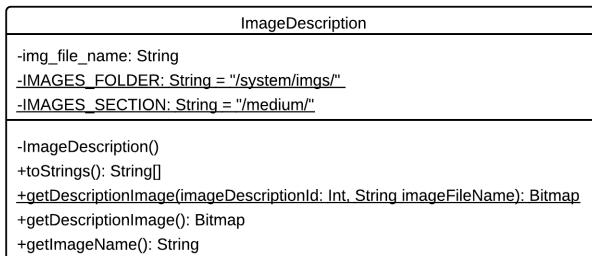


Figura 22: Classe com.woty.android.model.ImageDescription

Funzione della classe

Classe che rappresenta una descrizione con immagine di una quest del sistema e ne raccoglie le informazioni. Una quest che avrà associata una `ImageDescription` sarà una quest con immagine.

Relazioni con altre classi

- Estende la classe `com.woty.android.model.Description`

Attributi

- `String img_file_name`

Nome dell'immagine.

- `static final String IMAGES_FOLDER = "/system/imgs/"`

Stringa contenente l'indirizzo della cartella delle immagini delle quest da appendere all'indirizzo del server.

- `static final String IMAGES_SECTION = "/medium/"`

Stringa contenente l'indirizzo della sezione "mobile" delle immagini delle quest.

Metodi

- `ImageDescription()`

Viene definito il costruttore privato in quanto una `ImageDescription` è istanziabile solamente in ritorno da una richiesta al server.

+ `String[] toStrings()`

Metodo ridefinito dalla super classe che ritorna una stringa rappresentante l'oggetto.

```
+ static final Bitmap getDescriptionImage(int imageDescriptionId, String fileName)
    Metodo che ottiene l'immagine della descrizione dal server.

+ final Bitmap getDescriptionImage() throws NotLoggedException
    Metodo che ritorna il bitmap dell'immagine della descrizione.

+ String getImageName()
    Metodo getter dell'attributo img_file_name.
```

5.3.11 com.woty.android.model.Quest

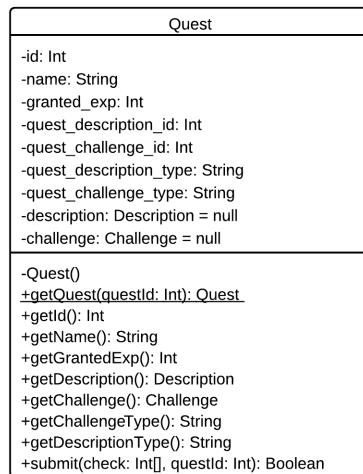


Figura 23: Classe com.woty.android.model.Quest

Funzione della classe

Classe che rappresenta una quest generica del sistema e ne raccoglie le informazioni.

Relazioni con altre classi

- Estende la classe com.woty.android.model.Resource
- Utilizza le classi:
 - com.woty.android.model.Challenge
 - com.woty.android.model.Description
 - com.woty.android.util.JParser
 - com.woty.android.exceptions.NotLoggedException

Attributi

- **int id**
 Identificativo della quest.
- **String name**
 Nome della quest.
- **int granted_exp**
 Numero di "punti esperienza" assegnati alla corretta risoluzione della quest.
- **int quest_description_id**
 Identificativo della descrizione associata alla quest.

- `quest_challenge_id`
Identificativo della sfida associata alla quest.
- `String quest_description_type`
Tipo della descrizione associata alla quest.
- `String quest_challenge_type`
Tipo della sfida associata alla quest.
- `Description description = null`
Oggetto `Description` che rappresenta la descrizione specifica della quest.
- `Challenge challenge = null`
Oggetto `Challenge` che rappresenta la sfida offerta dalla quest associato alla stessa.

Metodi

- `Quest()`
Viene definito il costruttore privato in quanto una `Quest` è istanziabile solamente in ritorno da una richiesta al server.
- + `static Quest getQuest(int questId) throws NotLoggedException`
Metodo che ritorna una quest dal server fornendo relativo id.
- + `int getId()`
Metodo *getter* dell'identificativo della quest.
- + `String getName()`
Metodo *getter* del nome della quest.
- + `int getGrantedExp()`
Metodo *getter* dei punti associati alla quest.
- + `Description getDescription() throws NotLoggedException`
Metodo *getter* dell'oggetto `Description` associato alla quest.
- + `Challenge getChallenge() throws NotLoggedException`
Metodo *getter* dell'oggetto `Challenge` associato alla quest.
- + `String getChallengeType()`
Metodo *getter* dell'oggetto `quest_challenge_type` associato alla quest.
- + `String getDescriptionType()`
Metodo *getter* dell'oggetto `quest_description_type` associato alla quest.
- + `boolean submit(int[] check, int questId) throws NotLoggedException`
Metodo che invia al server la risposta alla quest effettuata da uno User.

5.3.12 com.woty.android.model.QuestHistory

QuestHistory
<pre> -id: Int -quest_id: Int -quest_name: String -quest_exp: Int </pre>
<pre> -QuestHistory() +getPendingQuest(questId: Int): QuestHistory[] +getId(): Int +getQuestId(): Int +getQuestName(): String +getQuestExp(): Int </pre>

Figura 24: Classe com.woty.android.model.QuestHistory

Funzione della classe

Classe che rappresenta una quest associata a un utente, da svolgere o già svolta e ne raccoglie le informazioni. I dati dell'utente vengono ricavati tramite la classe **Session**.

Relazioni con altre classi

- Estende la classe `com.woty.android.model.Resource`
- Utilizza le classi:
 - `com.woty.android.model.Session`
 - `com.woty.android.util.JParser`
 - `com.woty.android.exceptions.NotLoggedException`

Attributi

- `int id`
Identificativo dell'oggetto.
- `int quest_id`
Identificativo della quest associata all'oggetto.
- `String quest_name`
Nome della quest associata all'oggetto.
- `int quest_exp`
"Punti esperienza" associati della quest associata all'oggetto.

Metodi

- `QuestHistory()`
Viene definito il costruttore privato in quanto la classe è istanziabile solamente in ritorno da una richiesta al server.
- + `static final QuestHistory[] getPendingQuest() throws NotLoggedException`
Metodo che ottiene un array di quest da svolgere dal server.
- + `int getId()`
Metodo *getter* dell'attributo `id`.
- + `int getQuestId()`
Metodo *getter* dell'attributo `quest_id`.
- + `String getQuestName()`
Metodo *getter* dell'attributo `quest_name`.
- + `int getQuestExp()`
Metodo *getter* dell'attributo `quest_exp`.

5.3.13 com.woty.android.model.Rank

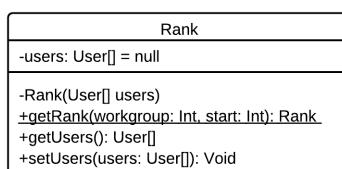


Figura 25: Classe com.woty.android.model.Rank

Funzione della classe

Classe che rappresenta una classifica del sistema e ne raccoglie le informazioni.

Relazioni con altre classi

- Estende la classe `com.woty.android.model.Resource`
- Utilizza le classi:
 - `com.woty.android.model.User`
 - `com.woty.android.util.JParser`
 - `com.woty.android.exceptions.InvalidArgumentException`
 - `com.woty.android.exceptions.NotLoggedException`
- È utilizzata dalle classi:
 - `com.woty.android.model.Session`

Attributi

- `User[] users = null`
Lista di User che fanno parte della classifica.

Metodi

- `Rank(User[] users)`
Viene definito il costruttore privato in quanto una classifica è istanziabile solamente in ritorno dalla richiesta al server.
- + `static final Rank getRank(int workgroup, int start) throws NotLoggedException`
Metodo che ottiene un Rank, che consiste in un array di User , dal server fornendo identificativo del workgroup e posizione di inizio della classifica.
- + `User[] getUsers()`
Metodo *get* dell'array di User.
- + `setUsers(User[] users)`
Metodo *set* dell'array di User.

5.3.14 com.woty.android.model.Session

Session
<pre> -FILENAME: String = "session" -EMAIL: String = "email" -PASSWORD: String = "password" -REGID: String = "regid" -authenticated: Boolean = false -userToken: String = null -user: User = null -regId: Boolean = false -context: Context = null; -clearSessionData(): Void +saveSession(context: Context): Boolean +loadSession(context: Context): Boolean +eraseSession(context: Context): Boolean +authenticate(email: String, password: String, context: Context): Boolean +registrationIdReceived(): Void +setApplicationContext(context: Context): Boolean +isInitialized(): Boolean +isAuthenticated(): Boolean +getUser(): User +getUserId(): Int +getUserToken(): String +setUser(u: User): Void </pre>

Figura 26: Classe com.woty.android.model.Session

Funzione della classe

Classe astratta che rappresenta e gestisce la sessione dell'utente nell'applicazione. Mantiene l'utente correttamente loggato, salva il token assegnatogli dal server e salva sul dispositivo le credenziali di accesso per evitare che l'utente debba reinserire i propri dati ad ogni apertura dell'applicazione. A questo scopo viene utilizzata la classe `android.content.SharedPreferences`. Al log-out dell'utente viene "pulita" la sessione e vengono eliminati i file dalla memoria del dispositivo.

Relazioni con altre classi

- Utilizza le classi:
 - `android.content.SharedPreferences`
 - `com.woty.android.model.User`
 - `com.woty.android.exceptions.NotLoggedException`
- È utilizzata dalle classi:
 - `com.woty.android.presenter.LoginP`
 - `com.woty.android.presenter.HomeP`
 - `com.woty.android.presenter.ModProfileP`
 - `com.woty.android.presenter.RanksP`
 - `com.woty.android.presenter.PlayerProfileP`
 - `com.woty.android.presenter.WaitingQuestP`
 - `com.woty.android.presenter.SolveQuestP`

Attributi

- `static final String FILENAME = "session"`
 Contiene il nome del file `SharedPreferences`.

- static final String EMAIL = "email"
Contiene il nome dell'attributo contenente l'indirizzo email dell'utente.
- static final String PASSWORD = "password"
Contiene il nome dell'attributo contenente la password dell'utente.
- static final String REGID = "regid"
Contiene il nome dell'attributo contenente il flag per il *registration id* al servizio C2DM.
- static boolean authenticated = false;
Campo che rappresenta se la sessione è autenticata oppure no.
- static String userToken = null
Campo che contiene il token dell'utente autenticato nel sistema.
- static User user = null
Campo che contiene l'utente loggato.
- static boolean regid = false
Campo booleano che indica se il *registration id* al servizio C2DM è impostato per l'utente corrente nel suo dispositivo.
- static Context context = null
Contesto dell'applicazione.

Metodi

- static void clearSessionData()
Metodo che elimina i dati della sessione.
- static boolean saveSession()
Metodo che salva i dati della sessione in un file `SharedPreferences` controllando se la sessione è autenticata.
- + static boolean loadSession()
Metodo che carica l'ultima sessione autenticata dal file salvato nel dispositivo. Sarà invocato all'apertura dell'applicazione permettendo il login automatico dell'utente.
- + static boolean eraseSession()
Metodo che elimina tutti i dati contenuti nel file salvato nel dispositivo. Sarà invocato qualora un utente effettui il log-out dal sistema.
- + static boolean authenticate(String email, String password)
Metodo che effettua l'autenticazione della sessione effettuando il login dell'utente e salvando i relativi dati nel file `SharedPreferences`. Viene invocato al primo login dell'utente per il salvataggio della sessione e dal metodo `loadSession()` nei login successivi.
- + static final void registrationIdReceived()
Metodo che aggiorna il *registration id* per il servizio C2DM una volta che questo è stato ricevuto.
- + static boolean setApplicationContext(Context context)
Metodo che imposta il contesto dell'applicazione.
- + static boolean isInitialized()
Metodo che dice se la sessione è inizializzata.
- + static boolean isAuthenticated()
Metodo che dice se la sessione è autenticata.
- + static User getUser()
Metodo che ritorna il riferimento all'utente autenticato.

+ static int getUserId()

Metodo che ritorna il numero identificativo dell'utente autenticato.

+ static String getUserToken()

Metodo che ritorna il token assegnato all'utente autenticato.

+ static boolean setUser(User u)

Metodo che aggiorna l'utente autenticato.

5.4 Specifica componente View

Componente che contiene tutte le viste e gli aspetti grafici dell'applicativo. Corrisponde al componente View del pattern MVP, riceve le richieste dall'utente e notifica le operazioni da eseguire al Presenter.

5.4.1 com.woty.android.view.Vista

Vista
#presenter: Presenter = null
#manageNotLoggedException(): Void
+startActivity(class: Class<?>): Void
+startActivity(context: Context, class: Class<?>, requestCode: Int): Void
+startActivityClearTop(class: Class<?>): Void
+hideKey(view: View, context: Context): Void
+toastString(str: String, context: Context): Void
+onCreateOptionsMenu(menu: Menu): Boolean
+onOptionsItemSelected(item: MenuItem): Boolean

Figura 27: Classe com.woty.android.view.Vista

Funzione della classe

Classe astratta che raccoglie attributi e metodi statici utilizzati dalle classi del componente View e che sarà estesa dalle stesse.

Relazioni con altre classi

- Estende la classe `android.app.Activity`.
- È estesa dalle classi:
 - `com.woty.android.view.Login`
 - `com.woty.android.view.Home`
 - `com.woty.android.view.ModProfile`
 - `com.woty.android.view.Ranks`
 - `com.woty.android.view.PlayerProfile`
 - `com.woty.android.view.PendingQuests`
 - `com.woty.android.view.SolveQuest`
- Utilizza le classi:
 - `com.woty.android.presenter.Presenter`
 - `com.woty.android.view.Help`
 - `com.woty.android.view.About`
 - `com.woty.android.exceptions.NotLoggedException`

Attributi

`# Presenter presenter = null`
Riferimento al presenter della classe.

Metodi

```
# final void manageNotLoggedException()
```

Metodo che gestisce l'eccezione `NotLoggedException` provocando il ritorno alla schermata di login.

```
+ void startActivityForResult(Class<?> class)
```

Metodo che avvia una nuova Activity.

```
+ void startActivityForResult(Context context, Class<?> class, int requestCode)
```

Metodo che avvia una nuova Activity attendendo un risultato tramite un `requestCode`.

```
+ void startActivityForResultClearTop(Class<?> class)
```

Metodo che avvia una nuova Activity e svuota lo stack delle stesse.

```
+ static void hideKey(View view, Context context)
```

Metodo invocato per impedire l'uscita automatica della tastiera a video.

```
+ static void toastString(String str, Context context)
```

Metodo invocato per la stampa a video della stringa `str` passata.

```
+ boolean onCreateOptionsMenu(Menu menu)
```

Metodo ridefinito dalla classe `android.app.Activity` richiamato ogni volta che viene premuto il bottone del menu dal dispositivo android. Si occupa dell'apertura del menu delle opzioni definito da `R.menu.optionsmenu`.

```
+ boolean onOptionsItemSelected(MenuItem item)
```

Metodo ridefinito dalla classe `android.app.Activity` invocato ogni volta che viene premuto un pulsante del menu delle opzioni. Si occupa di gestire le funzionalità offerte dal menu aprendo le relative finestre pop-up `About` e `Help` o effettuando il log-out dall'applicazione alla pressione dei relativi pulsanti.

5.4.2 com.woty.android.view.Login

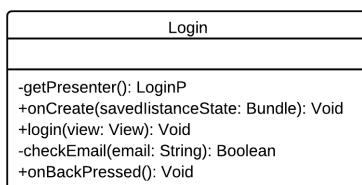


Figura 28: Classe com.woty.android.view.Login

Funzione della classe

Classe che si occupa della visualizzazione della finestra di login dalla quale l'utente può inserire le sue credenziali (email e password) per autenticarsi nel sistema Woty.

Relazioni con altre classi

- Estende la classe `com.woty.android.view.Vista`.
- Utilizza le classi:
 - `com.woty.android.presenter.LoginP`
 - `com.woty.android.exceptions.InvalidArgumentException`
- È utilizzata dalle classi:
 - `com.woty.android.presenter.LoginP`

Metodi

+ void **onCreate(Bundle savedInstanceState)**

Metodo ridefinito dalla classe `android.app.Activity` richiamato alla creazione della finestra. Al suo interno viene settato il layout `R.layout.login` e istanziato un nuovo oggetto `LoginP`.

- final LoginP **getPresenter()**

Metodo che ritorna ed effettua il *cast* dell'oggetto `presenter`.

+ void **login(View view)**

Metodo richiamato alla pressione del bottone "Login" che elabora i dati inseriti dall'utente e effettua l'accesso al sistema utilizzando la funzione `Session.authenticate`, o restituisce un messaggio di errore nel caso di login fallito.

- boolean **checkEmail(String email)**

Metodo che controlla se l'indirizzo email inserito dall'utente ha formato corretto.

+ void **onBackPressed()**

Metodo ridefinito per disabilitare il bottone "back" del dispositivo.

5.4.3 com.woty.android.view.Home

Home
<code>-getPresenter(): HomeP</code> <code>+onCreate(savedInstanceState: Bundle): Void</code> <code>+showProfile(): Void</code> <code>+modProfile(view: View): Void</code> <code>+startQuest(view: View): Void</code> <code>+startRank(view: View): Void</code> <code>#onResume(): Void</code>

Figura 29: Classe com.woty.android.view.Home

Funzione della classe

Classe che si occupa della visualizzazione della finestra contenente i dati del profilo dell'utente. Avvenuto l'accesso al sistema questa è la pagina principale del sistema e permette la modifica del profilo e l'accesso alle quest e alle classifiche.

Relazioni con altre classi

- Estende la classe `com.woty.android.view.Vista`.

- Utilizza le classi:

- `com.woty.android.view.ModProfile`
- `com.woty.android.view.PendingQuests`
- `com.woty.android.view.Ranks`
- `com.woty.android.presenter.HomeP`
- `com.woty.android.exceptions.InvalidArgumentException`
- `com.woty.android.exceptions.NotLoggedException`

- È utilizzata dalle classi:

- `com.woty.android.presenter.HomeP`

Metodi

+ void onCreate(Bundle savedInstanceState)

Metodo che viene richiamato alla creazione della finestra. Al suo interno viene settato il layout R.layout.home e istanziato un nuovo oggetto HomeP.

- final HomeP getPresenter()

Metodo che ritorna ed effettua il *cast* dell'oggetto presenter.

+ void showProfile() throws NotLoggedException

Metodo che carica i dati dell'utente e li visualizza nelle TextView della finestra.

+ void modProfile(View view)

Metodo che viene richiamato al click del bottone "Modifica Profilo" della finestra. Si occupa di avviare la finestra di modifica del profilo.

+ void startQuest(View view)

Metodo che viene richiamato al click del bottone "Svolgi quest" della finestra. Si occupa di avviare la finestra delle attività relative alle quest.

+ void startRank(View view)

Metodo che viene richiamato al click del bottone "Visualizza Classifiche" della finestra. Si occupa di avviare la finestra delle attività relative alle classifiche.

void onResume()

Metodo ridefinito dalla classe android.app.Activity invocato ogni volta in si ritorna a questa finestra per gestire un'eventuale eccezione di utente non loggato.

5.4.4 com.woty.android.view.ModProfile

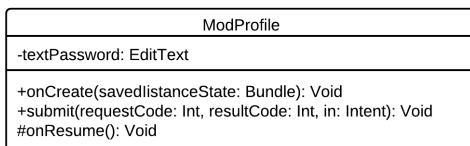


Figura 30: Classe com.woty.android.view.ModProfile

Funzione della classe

Classe che si occupa della visualizzazione della finestra che permette all'utente la modifica e il salvataggio dei propri dati personali modificabili.

Relazioni con altre classi

- Estende la classe com.woty.android.view.Vista.
- Utilizza le classi:
 - com.woty.android.presenter.ModProfileP
 - com.woty.android.exceptions.NotLoggedException
 - com.woty.android.exceptions.InvalidArgumentException
- È utilizzata dalle classi:
 - com.woty.android.view.Home
 - com.woty.android.presenter.ModProfileP

Attributi

- `EditText textPassword`

Casella di testo utilizzata dall'utente per modificare la propria password.

Metodi

- `final ModProfileP getPresenter()`

Metodo che ritorna ed effettua il *cast* dell'oggetto `presenter`.

+ `void onCreate(Bundle savedInstanceState)`

Metodo che viene richiamato alla creazione della finestra. Al suo interno viene settato il layout `R.layout.modprofile`.

+ `void submit(View view) throws NotLoggedException`

Metodo invocato alla pressione del bottone "Conferma" nella schermata. Provoca l'effettiva modifica dei dati dell'utente.

`void onResume()`

Metodo ridefinito dalla classe `android.app.Activity` richiamato ogni volta che la finestra diventa visibile all'utente. Si deve occupare dell'aggiornamento dei dati dell'utente contenuti dalle caselle di testo.

5.4.5 com.woty.android.view.PendingQuests

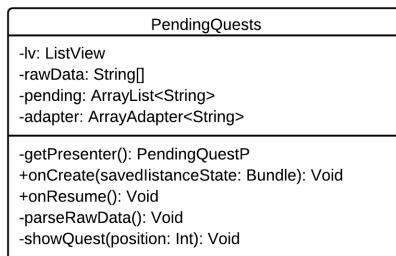


Figura 31: Classe com.woty.android.view.PendingQuests

Funzione della classe

Classe che si occupa della visualizzazione della schermata con la lista delle quest disponibili allo User.

Relazioni con altre classi

- Estende la classe `com.woty.android.view.Vista`.

- Utilizza le classi:

- `com.woty.android.view.SolveQuest`
- `com.woty.android.presenter.PendingQuestsP`
- `com.woty.android.exceptions.InvalidArgumentException`
- `com.woty.android.exceptions.NotLoggedException`

- È utilizzata dalle classi:

- `com.woty.android.view.Home`
- `com.woty.android.presenter.PendingQuestsP`

Attributi

- `ListView lv`
 Lista che espone le quest disponibili all'utente.
- `ArrayList<String[]> rawData`
`ArrayList` che contiene i dati "grezzi" riguardanti le quest da svolgere che arrivano dal server. Ogni elemento corrisponde ad una quest e contiene un array con i dati della stessa.
- `ArrayList<String> pending = new ArrayList<String>()`
`ArrayList` che contiene i dati utilizzabili nella lista di quest, ottenuti concatenando le stringhe per ogni elemento di `rawData`. Viene utilizzato in quanto un `ArrayAdapter` non funziona con array di array.
- `ArrayAdapter<String> adapter`
 `ArrayAdapter` della lista delle quest.

Metodi

- + `void onCreate(Bundle savedInstanceState)`
 Metodo che viene richiamato alla creazione della finestra. Al suo interno viene settato il layout `R.layout.pending_quest` e istanziato un nuovo oggetto `PendingQuestP`.
- `final PendingQuestP getPresenter()`
 Metodo che ritorna ed effettua il *cast* dell'oggetto `presenter`.
- # `void onResume()`
 Metodo ridefinito dalla classe `android.app.Activity` richiamato ogni volta che la finestra diventa visibile all'utente. Si deve occupare dell'aggiornamento della lista delle quest.
- `void showQuest(int position)`
 Metodo che provoca l'esecuzione di una quest aprendo una schermata `SolveQuest` alla selezione di una quest in posizione `position` nella lista da parte dell'utente.
- `final void parseRawData()`
 Metodo che effettua la concatenazione delle stringhe presenti nell'array `rawData` e setta così l'array `pending`.

5.4.6 com.woty.android.view.SolveQuest

SolveQuest	
-tvLabelChallenge: TextView	
-llDescription: LinearLayout	
-llChallenge: LinearLayout	
-llTrueFalse: LinearLayout	
-llTrueFalseLabel: LinearLayout	
-cb: CheckBox	
-rg: RadioGroup	
-rb: RadioButton	
-quest: ArrayList<String[]>	
-getPresenter(): SolveQuestP	
+onCreate(savedInstanceState: Bundle): Void	
-setUp(): Void	
+onResume(): Void	
-showQuest(questId: Int): Void	
+confirm(view: View): Void	

Figura 32: Classe com.woty.android.view.SolveQuest

Funzione della classe

Classe che si occupa della visualizzazione della schermata di risoluzione di una generica quest.

Relazioni con altre classi

- Estende la classe `com.woty.android.view.Vista`.
- Utilizza le classi:
 - `com.woty.android.presenter.SolveQuestP`
 - `com.woty.android.exceptions.InvalidArgumentException`
 - `com.woty.android.exceptions.NotLoggedException`
- È utilizzata dalle classi:
 - `com.woty.android.view.PendingQuests`
 - `com.woty.android.presenter.SolveQuestP`

Attributi

- **`TextView tvLabelChallenge`**
`TextView` che mostra a video la domanda associata alla sfida corrente.
- **`LinearLayout llDescription`**
Layout che contiene la descrizione testuale o con immagine della quest corrente.
- **`LinearLayout llChallenge`**
Layout che contiene la sfida corrente.
- **`LinearLayout llTrueFalse`**
Layout utilizzato per le quest con domande vero o falso.
- **`LinearLayout llTrueFalseLabel`**
Layout utilizzato per contenere le due risposte di una domanda vero/falso.
- **`CheckBox cb`**
`CheckBox` utilizzato nelle domande a risposta multipla.
- **`RadioGroup rg`**
`RadioGroup` utilizzato nella domande vero/falso o nelle domande a risposta singola tra un pool di possibili risposte.
- **`RadioButton rb`**
Riferimento al `RadioButton` selezionato dall'utente nel `RadioGroup` delle risposte.
- **`ArrayList<String[]> quest`**
`ArrayList<String[]>` che contiene le informazioni delle quest da svolgere.

Metodi

- + **`void onCreate(Bundle savedInstanceState)`**
Metodo che viene richiamato alla creazione della finestra. Al suo interno viene settato il layout `R.layout.solve_quest` e istanziato un nuovo oggetto `SolveQuestP`.
- **`SolveQuestP getPresenter()`**
Metodo che ritorna ed effettua il *cast* dell'oggetto `presenter`.
- **`void setUp()`**
Metodo che inizializza gli elementi della vista caricando gli id del layout xml.
- + **`void onResume()`**
Metodo ridefinito dalla classe `android.app.Activity` richiamato ogni volta che la finestra diventa visibile all'utente. Si occupa di richiedere al server le informazioni della quest in svolgimento.

- `showQuest(int questId) throws NotLoggedException`

Metodo che setta i valori degli elementi di vista della finestra con le informazioni della quest in svolgimento, caricata utilizzando il metodo `getQuestToSolve(questId)` del presenter associato.

+ `void confirm(View view) throws NotLoggedException`

Metodo invocato alla pressione del bottone "Conferma" della finestra, carica le risposte e le invia al server attraverso il presenter associato.

5.4.7 com.woty.android.view.Ranks

Ranks
<pre> -btnPrevious: Button -btnNext: Button -lv: ListView -rankFlag: Int = 0 -workgroupId: Int = 1 -playersPerRank: Int = 20 -loggedUserId: String = null -rawData: ArrayList<String> -players: ArrayList<String> -adapter: ArrayAdapter<String> -getPresenter(): RanksP +onCreate(savedInstanceState: Bundle): Void -resetActivity(): Void -parseRawData(): Void +myPosition(view: View): Void +showPlayerProfile(view View): Void +previousPlayers(view View): Void +nextPlayers(view View): Void </pre>

Figura 33: Classe com.woty.android.view.Ranks

Funzione della classe

Classe che si occupa della visualizzazione della finestra che espone tutte le classifiche definite per la gamification del sistema. Per limitare la richiesta di dati vengono visualizzati solo 20 utenti alla volta nella schermata e viene offerta la possibilità di caricarne altri con la pressione di un apposito pulsante. Da questa pagina è inoltre possibile visualizzare le statistiche degli utenti presenti nelle varie classifiche.

Relazioni con altre classi

- Estende la classe `com.woty.android.view.Vista`.
- Utilizza le classi:
 - `com.woty.android.view.PlayerProfile`
 - `com.woty.android.presenter.RanksP`
 - `com.woty.android.exceptions.InvalidArgumentException`
 - `com.woty.android.exceptions.NotLoggedException`
- È utilizzata dalle classi:
 - `com.woty.android.view.Home`
 - `com.woty.android.presenter.RanksP`

Attributi

- **Button btnPrevious**
Pulsante che carica i punteggi della classifica per i `playersPerRank` utenti precedenti nella schermata.
- **Button btnNext**
Pulsante che carica i punteggi della classifica per i `playersPerRank` utenti successivi nella schermata.
- **ListView lv**
Lista che espone gli utenti visualizzati nella classifica.
- **int rankFlag = 0**
Intero che indica la posizione corrente del primo giocatore visualizzato nella classifica.
- **int playersPerRank = 20**
Intero che indica il numero massimo di giocatori visualizzabili nella listview.
- **int workgroupId = 1**
Identificativo del workgroup collegato alla classifica.
- **String loggedUserId = null**
Contiene l'identificativo del giocatore loggato nel sistema, che verrà utilizzato per evidenziare l'utente corrente nella classifica.
- **ArrayList<String[]> rawData = new ArrayList<String[]>()**
`ArrayList` che contiene i dati "grezzi" riguardanti gli utenti presenti nella classifica che arrivano dal server. Ogni elemento corrisponde ad uno User e contiene un array con i dati dello stesso.
- **ArrayList<String> players = new ArrayList<String>()**
`ArrayList` che contiene i dati utilizzabili nella lista dei giocatori, ottenuti concatenando le stringhe per ogni elemento di `rawData`. Viene utilizzato in quanto un `ArrayAdapter` non funziona con array di array.
- **ArrayAdapter<String> adapter**
 `ArrayAdapter` della lista delle quest.

Metodi

- + **void onCreate(Bundle savedInstanceState)**
Metodo che viene richiamato alla creazione della finestra. Al suo interno viene settato il layout `R.layout.rank` e vengono caricati i pulsanti e la classifica settando i relativi comportamenti al loro click.
- **final RanksP getPresenter()**
Metodo che ritorna ed effettua il *cast* dell'oggetto `presenter`.
- **void resetActivity() throws NotLoggedException**
Metodo che provoca il *reset* dell'Activity facendo tornare tutti i suoi attributi allo stato iniziale.
- **final void parseRawData()**
Metodo che effettua la concatenazione delle stringhe presenti nell'array `rawData` e setta così l'array `players`.
- + **void myPosition(View view)**
Metodo utilizzato per visualizzare nella lista la zona di classifica contenente l'utente corrente.

+ void showPlayerProfile(int position)

Metodo invocato quando l'utente seleziona un item della classifica, provocando l'apertura di una finestra PlayerProfile con i dati del giocatore selezionato.

+ void nextPlayers(View view)

Metodo invocato alla pressione del pulsante **btnNext** che provoca la richiesta al server dei 20 giocatori successivi nella classifica, sovrascrivendo l'array **rawData**, e l'aggiornamento della lista con i nuovi valori.

void previousPlayers(View view)

Metodo invocato alla pressione del pulsante **btnPrevious** che provoca la richiesta al server dei 20 giocatori precedenti nella classifica, sovrascrivendo l'array **rawData**, e l'aggiornamento della lista con i nuovi valori.

5.4.8 com.woty.android.view.PlayerProfile

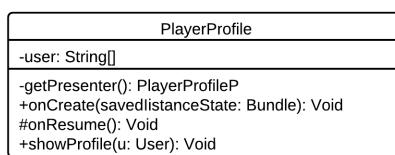


Figura 34: Classe com.woty.android.view.PlayerProfile

Funzione della classe

Classe che si occupa della visualizzazione della finestra contenente i dati relativi ad un utente selezionato all'interno di una classifica.

Relazioni con altre classi

- Estende la classe `com.woty.android.view.Vista`.
- Utilizza le classi:
 - `com.woty.android.presenter.PlayerProfileP`
 - `com.woty.android.exceptions.InvalidArgumentException`
 - `com.woty.android.exceptions.NotLoggedException`
- È utilizzata dalle classi:
 - `com.woty.android.view.Ranks`
 - `com.woty.android.presenter.PlayerProfileP`

Attributi

- User targetUser

Riferimento all'utente del quale si vogliono vedere il profilo e le statistiche.

Metodi

+ void onCreate(Bundle savedInstanceState)

Metodo che viene richiamato alla creazione della finestra. Al suo interno viene settato il layout `R.layout.player_profile`.

- void showProfile(User u)

Metodo che carica nella schermata i dati dell'utente da visualizzare.

```
# void onResume()
```

Metodo ridefinito dalla classe `android.app.Activity` richiamato ogni volta che la finestra diventa visibile all'utente. Si occupa di richiedere al server le statistiche aggiornate dell'utente da visualizzare.

5.4.9 com.woty.android.view.Help

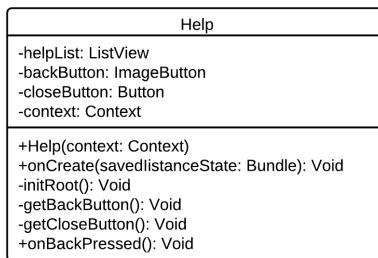


Figura 35: Classe com.woty.android.view.Help

Funzione della classe

La funzione di questa classe è la visualizzazione della finestra della guida del prodotto al click del relativo pulsante nel menu delle opzioni.

Relazioni con altre classi

- Estende la classe `android.app.Dialog`.
- È utilizzata dalle classi:
 - `com.woty.android.view.Vista`
 - `com.woty.android.view.Home`
 - `com.woty.android.view.ModProfile`
 - `com.woty.android.view.PlayerProfile`
 - `com.woty.android.view.RankActivity`
 - `com.woty.android.view.PendingQuests`
 - `com.woty.android.view.SolveQuest`

Attributi

- `ListView helpList`
Lista che elenca le sezioni della guida.
- `ImageButton backButton`
Bottone che permette il ritorno alla pagina iniziale della guida.
- `Button closeButton`
Bottone che chiude la guida.

Metodi

- + `Help(Context context)`
Costruttore pubblico della classe.
- + `void onCreate(Bundle savedInstanceState)`
Metodo che viene richiamato alla creazione della finestra di dialogo.

- void initRoot()

Metodo che imposta il layout della finestra a R.layout.helpRoot e imposta i comportamenti della finestra al click delle varie sezioni, cambiandone dinamicamente il layout.

- void getBackButton()

Metodo che si occupa dell'aggiornamento della variabile backButton ad ogni cambio di layout e imposta il comportamento al click del bottone.

- void getCloseButton()

Metodo che si occupa dell'aggiornamento della variabile closeButton ad ogni cambio di layout e imposta il comportamento al click del bottone.

+ void onBackPressed()

Metodo ridefinito da android.app.Activity per modificare il comportamento della finestra di dialogo alla pressione del pulsante "Back" dei dispositivi Android. Ridefinito per ritornare alla pagina principale della guida nel caso in cui l'utente sia in una delle sottosezioni, evitando la chiusura della finestra di dialogo.

5.4.10 com.woty.android.view.About

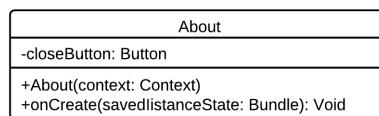


Figura 36: Classe com.woty.android.view.About

Funzione della classe

Classe che si occupa della visualizzazione della finestra delle informazioni sul prodotto al click del relativo pulsante nel menu delle opzioni.

Relazioni con altre classi

- Estende la classe android.app.Dialog.
- È utilizzata dalle classi:
 - com.woty.android.view.Vista
 - com.woty.android.view.Home
 - com.woty.android.view.ModProfile
 - com.woty.android.view.PlayerProfile
 - com.woty.android.view.RankActivity
 - com.woty.android.view.SolveQuest

Attributi

- Button closeButton

Bottone che chiude la finestra di dialogo.

Metodi

+ `About(Context context)`

Costruttore pubblico della classe.

+ `void onCreate(Bundle savedInstanceState)`

Metodo che viene richiamato alla creazione della finestra di dialogo.

Imposta il layout della finestra a `R.layout.about` e imposta l'azione del bottone di chiusura.

5.5 Specifica componente Presenter

Rappresenta il componente Presenter del pattern MVP; risiede tra il componente Model e il componente View, il suo scopo è quello di inoltrare richieste provenienti da una view generica al model e notificare eventuali cambiamenti di stato di quest'ultimo.

5.5.1 com.woty.android.presenter.Presenter



Figura 37: Classe com.woty.android.presenter.Presenter

Funzione della classe

Classe astratta che rappresenta un generico oggetto presenter e contiene attributi e metodi condivisi da tutte le sue sottoclassi.

Relazioni con altre classi

- Utilizza le classi:
 - com.woty.android.model.Session
 - com.woty.android.view.Vista
 - com.woty.android.exceptions.NotLoggedException
 - com.woty.android.exceptions.InvalidArgumentException
- È estesa dalle classi:
 - com.woty.android.presenter.LoginP
 - com.woty.android.presenter.HomeP
 - com.woty.android.presenter.PendingQuestsP
 - com.woty.android.presenter.SolveQuestP
 - com.woty.android.presenter.RanksP
 - com.woty.android.presenter.PlayerProfileP
 - com.woty.android.presenter.ModProfileP

Attributi

Vista vista = null

Riferimento al generico oggetto del componente View per la quale la classe che estenderà Presenter fungerà da presenter specifico.

Metodi

Presenter(Vista _vista) throws InvalidArgumentException

Costruttore protetto della classe che riceve e setta la vista associata al presenter.

+ static final boolean logout() throws NotLoggedException

Metodo che effettua il log out dello User e cancella la sessione corrente.

+ static final boolean setUserRegId(String regId) throws NotLoggedException

Metodo che provoca la modifica nel server del *registration id* per il servizio C2DM dell'utente.

5.5.2 com.woty.android.presenter.LoginP

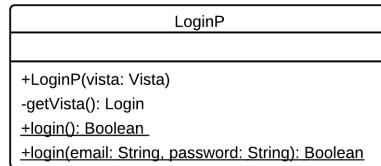


Figura 38: Classe com.woty.android.presenter.LoginP

Funzione della classe

Offre le funzionalità di presenter specifiche per la classe com.woty.android.view.Login.

Relazioni con altre classi

- Estende la classe com.woty.android.presenter.Presenter.
- Utilizza le classi:
 - com.woty.android.view.Vista
 - com.woty.android.view.Login
 - com.woty.android.model.User
 - com.woty.android.model.Session
 - com.woty.android.exceptions.NotLoggedException
 - com.woty.android.exceptions.InvalidArgumentException
- È utilizzata dalle classi:
 - com.woty.android.view.Login

Metodi

- + LoginP(Vista vista) throws InvalidArgumentException
 Costruttore della classe che riceve la vista collegata.
- final Login getVista()
 Metodo che ritorna ed effettua il *cast* della classe Login collegata al presenter.
- + static final boolean login()
 Metodo che carica la sessione precedente ed effettua il login automatico dell'utente.
- + static final boolean login(String email, String password)
 Metodo che effettua il login di un utente ricevendo email e password dello stesso.

5.5.3 com.woty.android.presenter.HomeP

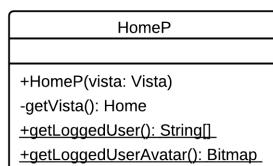


Figura 39: Classe com.woty.android.presenter.HomeP

Funzione della classe

Offre le funzionalità di presenter specifiche per la classe `com.woty.android.view.Home`.

Relazioni con altre classi

- Estende la classe `com.woty.android.presenter.Presenter`.
- Utilizza le classi:
 - `com.woty.android.view.Vista`
 - `com.woty.android.view.Home`
 - `com.woty.android.model.User`
 - `com.woty.android.exceptions.NotLoggedException`
 - `com.woty.android.exceptions.InvalidArgumentException`
- È utilizzata dalle classi:
 - `com.woty.android.view.Home`

Metodi

- + `HomeP(Vista vista) throws InvalidArgumentException`
Costruttore della classe che riceve la vista collegata.
- `final Home getView()`
Metodo che ritorna ed effettua il *cast* della classe `Home` collegata al presenter.
- + `static final String[] getLoggedUser() throws NotLoggedException`
Metodo che ottiene l'utente loggato nella sessione corrente.
- + `static final Bitmap getLoggedUserAvatar() throws NotLoggedException`
Metodo che ottiene l'avatar dell'utente loggato nella sessione corrente.

5.5.4 com.woty.android.presenter.PlayerProfileP

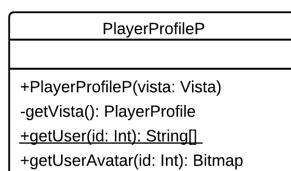


Figura 40: Classe `com.woty.android.presenter.PlayerProfileP`

Funzione della classe

Offre le funzionalità di presenter specifiche per la classe `com.woty.android.view.PlayerProfile`.

Relazioni con altre classi

- Estende la classe `com.woty.android.presenter.Presenter`.
- Utilizza le classi:
 - `com.woty.android.view.Vista`
 - `com.woty.android.view.PlayerProfile`
 - `com.woty.android.model.User`

- com.woty.android.exceptions.NotLoggedException
- com.woty.android.exceptions.InvalidArgumentException
- È utilizzata dalle classi:
 - com.woty.android.view.PlayerProfile

Metodi

- + PlayerProfileP(Vista vista) throws InvalidArgumentException
Costruttore della classe che riceve la vista collegata.
- final PlayerProfile getVista()
Metodo che ritorna ed effettua il *cast* della classe PlayerProfile collegata al presenter.
- + static final String[] getUser(int id) throws NotLoggedException
Metodo che ottiene le informazioni di un utente nel server passando il relativo id.
- + static final Bitmap getUserAvatar(int id) throws NotLoggedException
Metodo che ottiene l'avatar di un utente generico passando il relativo id.

5.5.5 com.woty.android.presenter.ModProfileP

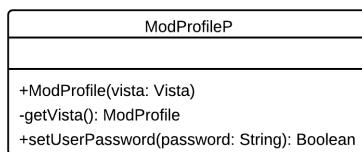


Figura 41: Classe com.woty.android.presenter.ModProfileP

Funzione della classe

Offre le funzionalità di presenter specifiche per la classe com.woty.android.view.ModProfile.

Relazioni con altre classi

- Estende la classe com.woty.android.presenter.Presenter.
- Utilizza le classi:
 - com.woty.android.view.Vista
 - com.woty.android.view.ModProfile
 - com.woty.android.model.User
 - com.woty.android.exceptions.NotLoggedException
 - com.woty.android.exceptions.InvalidArgumentException
- È utilizzata dalle classi:
 - com.woty.android.view.ModProfile

Metodi

- + ModProfileP(Vista vista) throws InvalidArgumentException
Costruttore della classe che riceve la vista collegata.
- final ModProfile getVista()
Metodo che ritorna ed effettua il *cast* della classe ModProfile collegata al presenter.
- + static final boolean setUserPassword(String password) throws NotLoggedException
Metodo che modifica la password dell'utente corrente.

5.5.6 com.woty.android.presenter.PendingQuestsP

PendingQuestsP
-qh: QuestHistory[] = null;
+PendingQuestsP(vista: Vista)
-getVista(): PendingQuests
+getPendingQuests(): String[]
+getQuestIdFromPosition(position: Int): Int
+setQuestHistory(): Void

Figura 42: Classe com.woty.android.presenter.PendingQuestsP

Funzione della classe

Offre le funzionalità di presenter specifiche per la classe com.woty.android.view.PendingQuests.

Relazioni con altre classi

- Estende la classe com.woty.android.presenter.Presenter.
- Utilizza le classi:
 - com.woty.android.view.Vista
 - com.woty.android.view.PendingQuests
 - com.woty.android.model.Quest
 - com.woty.android.model.QuestHistory
 - com.woty.android.model.Session
 - com.woty.android.exceptions.NotLoggedException
 - com.woty.android.exceptions.InvalidArgumentException
- È utilizzata dalle classi:
 - com.woty.android.view.PendingQuests

Attributi

- QuestHistory[] qh = null

Lista delle quest associate all'utente. Sono comprese le quest da effettuare e quelle già effettuate.

Metodi

- + PendingQuestsP(Vista vista) throws InvalidArgumentException

Costruttore della classe che riceve la vista associata.
- final PendingQuests getVista()

Metodo che ritorna ed effettua il cast della classe PendingQuests collegata al presenter.
- + String[] getPendingQuest() throws NotLoggedException

Metodo che ottiene le informazioni di una quest da svolgere nel server.
- + int getQuestIdFromPosition(int position) throws NotLoggedException

Metodo che ritorna l'identificativo della quest selezionata dall'utente nella lista data la posizione della stessa nella ListView.
- + void setQuestHistory() throws NotLoggedException

Metodo set della lista delle quest. L'attributo viene settato tramite l'utilizzo della funzione getPendingQuests() della classe QuestHistory.

5.5.7 com.woty.android.presenter.SolveQuestP

SolveQuestP
-quest: Quest
+SolveQuestP(vista: Vista)
-getVista(): SolveQuest
+getImage(questId: Int): Bitmap
+getQuestToSolve(questId: Int): ArrayList<String[]>

Figura 43: Classe com.woty.android.presenter.SolveQuestP

Funzione della classe

Offre le funzionalità di presenter specifiche per la classe com.woty.android.view.SolveQuest.

Relazioni con altre classi

- Estende la classe com.woty.android.presenter.Presenter.
- Utilizza le classi:
 - com.woty.android.model.Challenge
 - com.woty.android.model.ComboChallenge
 - com.woty.android.model.Description
 - com.woty.android.model.ImageDescription
 - com.woty.android.model.MultiOptionChallenge
 - com.woty.android.model.QRChallenge
 - com.woty.android.model.Quest
 - com.woty.android.model.Resource
 - com.woty.android.model.Session
 - com.woty.android.model.TextDescription
 - com.woty.android.model.TrueFalseChallenge
 - com.woty.android.model.User
 - com.woty.android.view.Vista
 - com.woty.android.view.SolveQuest
 - com.woty.android.exceptions.NotLoggedException
 - com.woty.android.exceptions.InvalidArgumentException
- È utilizzata dalle classi:
 - com.woty.android.view.SolveQuest

Attributi

- **Quest quest**
Riferimento alla quest che l'utente sta svolgendo.

Metodi

- **final SolveQuest getVista()**

Metodo che ritorna ed effettua il *cast* della classe **SolveQuest** collegata al presenter.

+ **final Bitmap getImage(int questId) throws NotLoggedException**

Metodo che ottiene l'immagine associata alla quest dal server fornendo l'identificativo della quest.

+ **boolean submit(int[] check) throws NotLoggedException**

Metodo che invia la risposta alla quest data dall'utente alla conferma della stessa.

+ **ArrayList<String[]> getQuestToSolve(int questId) throws NotLoggedException**

Metodo che ottiene i dati della quest che l'utente ha deciso di svolgere dal server e li ritorna in un **ArrayList<String[]>**.

5.5.8 com.woty.android.presenter.RanksP

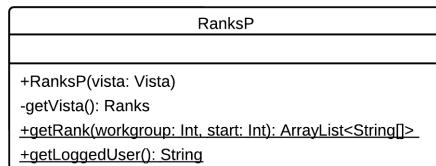


Figura 44: Classe com.woty.android.presenter.RanksP

Funzione della classe

Offre le funzionalità di presenter specifiche per la classe **com.woty.android.view.Ranks**.

Relazioni con altre classi

- Estende la classe **com.woty.android.presenter.Presenter**.

- Utilizza le classi:

- **com.woty.android.view.Vista**
- **com.woty.android.view.Ranks**
- **com.woty.android.model.User**
- **com.woty.android.model.Rank**
- **com.woty.android.exceptions.NotLoggedException**
- **com.woty.android.exceptions.InvalidArgumentException**

- È utilizzata dalle classi:

- **com.woty.android.view.Ranks**

Metodi

+ **RanksP(Vista vista) throws InvalidArgumentException**

Costruttore della classe che riceve la vista collegata.

- **final Ranks getVista()**

Metodo che ritorna ed effettua il *cast* della classe **Ranks** collegata al presenter.

```
+ static final ArrayList<String[]> getRank(int workgroup, int start) throws NotLoggedException
```

Metodo che ottiene dal server un Rank, sotto forma di array di stringhe contenenti le informazioni sugli utenti presenti in classifica, passando workgroup di appartenenza e posizione iniziale richiesta.

```
+ static final String getLoggedUser() throws NotLoggedException
```

Metodo che ottiene la stringa con l'id dell'utente loggato nella sessione corrente.

5.5.9 com.woty.android.presenter.C2dmMessageReceiver

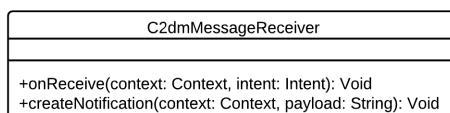


Figura 45: Classe com.woty.android.presenter.C2dmMessageReceiver

Funzione della classe

Classe che gestisce la ricezione delle notifiche dal server, tramite il servizio C2DM, per l'arrivo di nuove quest.

Relazioni con altre classi

- Estende la classe `android.content.BroadcastReceiver`.

Metodi

```
+ void onReceive(Context context, Intent intent)
```

Metodo che gestisce la ricezione del messaggio dal server Google.

```
+ void createNotification(Context context, String payload)
```

Metodo utilizzato per creare e visualizzare una notifica del messaggio ricevuto.

5.5.10 com.woty.android.presenter.C2dmRegistrationReceiver

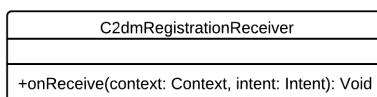


Figura 46: Classe com.woty.android.presenter.C2dmRegistrationReceiver

Funzione della classe

Classe che gestisce la visualizzazione della conferma di avvenuta registrazione dell'applicazione al servizio C2DM.

Relazioni con altre classi

- Estende la classe `android.content.BroadcastReceiver`.
- Utilizza le classi:

- `com.woty.android.model.Session`

- `com.woty.android.model.User`

- `com.woty.android.exceptions.NotLoggedException`

Metodi

+ void **onReceive(Context context, Intent intent)**

Metodo che gestisce la ricezione del *registration id* del servizio C2DM, salvandolo nella sessione dello User corrente in caso di successo o visualizzando una stringa di errore in caso contrario.

5.6 Specifica componente Util

Componente che offre le funzionalità per effettuare tutte le richieste al server Woty e si occupa inoltre della gestione del servizio C2DM per la ricezione delle notifiche sull'arrivo di nuove quest dedicate allo User.

5.6.1 com.woty.android.util.JParser

JParser
-parser: Gson
+parseJson(json: String, objClass: Class): Object +parseObject(obj: Object): String

Figura 47: Classe com.woty.android.remote.JParser

Funzione della classe

Classe astratta che fornisce le funzionalità di parser^g per il formato json. Per la definizione di questa classe ci si appoggia alla libreria com.google.gson.Gson.

Relazioni con altre classi

- Utilizza la classe:
 - com.google.gson.Gson
- È utilizzata dalle classi:
 - com.woty.android.model.User
 - com.woty.android.model.Rank

Attributi

- static final Gson parser = new Gson()
Istanza del parser json della libreria Google.

Metodi

- + static Object parseJson(String json, Class<?> objClass)
Metodo che converte una stringa json ricevuta dal server in un oggetto generico della classe objClass.
- + static String parseObject(Object obj)
Metodo che converte un oggetto in una stringa json da inviare al server.

5.7 Specifica package exceptions

Package che contiene tutte le eccezioni sollevabili dai metodi dell'applicazione.

5.7.1 com.woty.android.exceptions.NotLoggedException

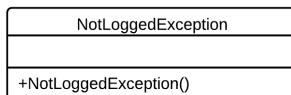


Figura 48: Classe com.woty.android.exceptions.NotLoggedException

Funzione della classe

Eccezione sollevata nel caso in cui non ci siano utenti loggati nel sistema.

Metodi

- + `NotLoggedException()`

Costruttore pubblico della classe.

5.7.2 com.woty.android.exceptions.InvalidArgumentException

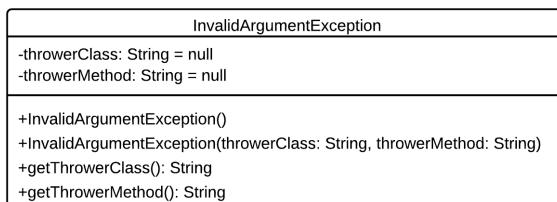


Figura 49: Classe com.woty.android.exceptions.InvalidArgumentException

Funzione della classe

Eccezione sollevata nel caso in cui si costruisce un presenter passando una vista non valida.

Attributi

- `String throwerClass = null`

Stringa contenente il nome della classe che ha sollevato l'eccezione.

- `String throwerMethod = null`

Stringa contenente il nome del metodo che ha sollevato l'eccezione.

Metodi

- + `InvalidArgumentException()`

Costruttore pubblico della classe.

- + `InvalidArgumentException(String _throwerClass, String _throwerMethod)`

Costruttore pubblico della classe a cui vengono passati classe e metodo che hanno sollevato l'eccezione.

- + `String getThrowerClass()`

Metodo *getter* dell'attributo `throwerClass`.

+ `String getThrowerMethod()`
Metodo *getter* dell'attributo `throwerMethod`.