



## Definizione Di Prodotto

### Informazioni sul documento

<b>Titolo documento</b>	Definizione Di Prodotto
<b>Versione attuale</b>	v2.0.0
<b>Data versione attuale</b>	2012/04/02
<b>Data creazione</b>	2012/02/15
<b>Redazione</b>	Stefano Faoro Giacomo Lorigiola Luca Guerra
<b>Revisione</b>	Umberto Dall'Est[v2.0.0] Antonio Pretto[v1.0.0]
<b>Approvazione</b>	Giacomo Lorigiola
<b>Stato documento</b>	Formale
<b>Uso</b>	Esterno
<b>Distribuito da</b>	SevenFold
<b>Destinato a</b>	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
v2.0.0	2012/04/02	Giacomo Lorigola	Approvazione e rilascio seconda versione
v1.5.0	2012/01/02	Stefano Faoro	Correzione figura 1
v1.4.0	2012/04/01	Luca Guerra	Inserimento tipi ritorno in attributi e metodi sottosistema server
v1.3.0	2012/04/01	Stefano Faoro	Varie correzioni ortografiche
v1.2.0	2012/03/30	Luca Guerra	Inserimento diagrammi sequenza sottosistema server
v1.1.0	2012/03/29	Stefano Faoro	Aggiornata sezione Mobile
v1.1.0	2012/03/27	Stefano Faoro	Aggiornati riferimenti normativi e uniformata struttura documento
v1.0.0	2012/04/02	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

<b>1</b>	<b>Introduzione</b>	<b>8</b>
1.1	Scopo del documento . . . . .	8
1.2	Scopo del prodotto . . . . .	8
1.3	Glossario . . . . .	8
1.4	Riferimenti . . . . .	8
1.4.1	Normativi . . . . .	8
1.4.2	Informativi . . . . .	8
<b>2</b>	<b>Standard di progetto</b>	<b>9</b>
2.1	Standard di progettazione architetturale . . . . .	9
2.2	Standard di documentazione del codice . . . . .	9
2.3	Standard di denominazione di entità e relazioni . . . . .	9
2.4	Standard di programmazione . . . . .	9
2.5	Strumenti di lavoro . . . . .	9
<b>3</b>	<b>Specifiche Woty Server</b>	<b>10</b>
3.1	Introduzione . . . . .	10
3.2	Wildcard . . . . .	10
3.3	Proprietà del linguaggio Ruby . . . . .	10
3.4	Implementazione standard di una risorsa . . . . .	10
3.4.1	Model . . . . .	11
3.4.2	Controller . . . . .	11
3.4.3	View . . . . .	12
3.4.4	Fogli di stile . . . . .	12
3.4.5	Javascript . . . . .	12
3.5	Comportamenti e necessità anomale . . . . .	12
3.5.1	Decorator . . . . .	12
3.5.2	Helper . . . . .	13
3.5.3	Gerarchie standard . . . . .	13
3.5.4	Classe ApplicationController . . . . .	13
3.5.5	Classe ActiveRecord::Base . . . . .	14
3.6	Specifiche risorse standard . . . . .	15
3.6.1	User . . . . .	15
3.6.2	SuperUser . . . . .	17
3.6.3	DesktopUser . . . . .	18
3.6.4	MobileUser . . . . .	18
3.6.5	Admin . . . . .	19
3.6.6	Customer . . . . .	20
3.6.7	WorkGroup . . . . .	21
3.6.8	Plan . . . . .	22
3.6.9	Quest . . . . .	23
3.6.10	QuestDescription . . . . .	25
3.6.11	TextDescription . . . . .	25
3.6.12	ImageDescription . . . . .	26
3.6.13	VideoDescription . . . . .	26
3.6.14	QuestChallenge . . . . .	26
3.6.15	ComboChallenge . . . . .	27
3.6.16	TrueFalseChallenge . . . . .	28
3.6.17	TrueFalseChallenge . . . . .	28
3.6.18	MultiOptionChallenge . . . . .	29
3.6.19	Achievement . . . . .	29
3.6.20	Like . . . . .	30
3.6.21	Comment . . . . .	31
3.6.22	StreamElement . . . . .	32

3.6.23	Specifica Classe QuestHistory . . . . .	32
3.6.24	Ticket . . . . .	33
3.6.25	CompletedAchievement . . . . .	34
3.6.26	CompletedAchievementElement . . . . .	35
3.6.27	Condition . . . . .	36
3.6.28	CustomQuest . . . . .	36
3.6.29	QuestAssociation . . . . .	37
3.6.30	QuestCategory . . . . .	37
3.6.31	SharedQuest . . . . .	38
3.6.32	SignupElement . . . . .	38
3.6.33	StatusChangeElement . . . . .	39
3.6.34	Wgcategorie . . . . .	39
3.7	Altre risorse . . . . .	40
3.7.1	XmlrpcController . . . . .	40
3.7.2	Autenticazione . . . . .	40
3.7.3	Autorizzazione . . . . .	41
3.7.4	PagesController . . . . .	42
3.7.5	Moduli . . . . .	43
<b>4</b>	<b>Specifiche Woty Desktop</b>	<b>44</b>
4.1	Diagramma Classi . . . . .	44
4.2	Specifiche componente Manager . . . . .	44
4.2.1	Specifiche classe Manager . . . . .	45
4.2.2	Specifiche classe Exception . . . . .	48
4.3	Specifiche componente Client . . . . .	49
4.3.1	Specifiche classe Client . . . . .	49
4.3.2	Specifiche libreria LibMaia . . . . .	50
4.4	Specifiche componente Account . . . . .	51
4.4.1	Specifiche classe LoginWindow . . . . .	51
4.4.2	Specifiche classe Account . . . . .	52
4.4.3	Specifiche libreria simpleCrypt . . . . .	54
4.4.4	Specifiche classe Preferences . . . . .	54
<b>5</b>	<b>Specifiche Woty Mobile</b>	<b>57</b>
5.1	Diagramma delle Classi . . . . .	57
5.2	Specifiche componente Model . . . . .	58
5.2.1	com.woty.android.model.Resource . . . . .	58
5.2.2	com.woty.android.model.Configuration . . . . .	59
5.2.3	com.woty.android.model.User . . . . .	61
5.2.4	com.woty.android.model.Workgroup . . . . .	63
5.2.5	com.woty.android.model.Challenge . . . . .	64
5.2.6	com.woty.android.model.TrueFalseChallenge . . . . .	66
5.2.7	com.woty.android.model.ComboChallenge . . . . .	66
5.2.8	com.woty.android.model.MultiOptionChallenge . . . . .	67
5.2.9	com.woty.android.model.QRChallenge . . . . .	67
5.2.10	com.woty.android.model.Description . . . . .	68
5.2.11	com.woty.android.model.TextDescription . . . . .	69
5.2.12	com.woty.android.model.ImageDescription . . . . .	69
5.2.13	com.woty.android.model.VideoDescription . . . . .	70
5.2.14	com.woty.android.model.Quest . . . . .	71
5.2.15	com.woty.android.model.QuestHistory . . . . .	72
5.2.16	com.woty.android.model.UsersRank . . . . .	73
5.2.17	com.woty.android.model.WorkgroupsRank . . . . .	75
5.2.18	com.woty.android.model.Session . . . . .	76
5.3	Specifiche componente View . . . . .	79
5.3.1	com.woty.android.view.Vista . . . . .	79

5.3.2	com.woty.android.view.Login . . . . .	80
5.3.3	com.woty.android.view.Home . . . . .	81
5.3.4	com.woty.android.view.ModProfile . . . . .	82
5.3.5	com.woty.android.view.PendingQuests . . . . .	83
5.3.6	com.woty.android.view.SolveQuest . . . . .	85
5.3.7	com.woty.android.view.WorkgroupsRanks . . . . .	87
5.3.8	com.woty.android.view.UsersRanks . . . . .	89
5.3.9	com.woty.android.view.RankAdapter . . . . .	91
5.3.10	com.woty.android.view.PlayerProfile . . . . .	91
5.3.11	com.woty.android.view.Help . . . . .	92
5.3.12	com.woty.android.view.About . . . . .	94
5.3.13	com.woty.android.view.ShowNotification . . . . .	94
5.4	Specifica componente Presenter . . . . .	96
5.4.1	com.woty.android.presenter.Presenter . . . . .	96
5.4.2	com.woty.android.presenter.LoginP . . . . .	97
5.4.3	com.woty.android.presenter.HomeP . . . . .	98
5.4.4	com.woty.android.presenter.PlayerProfileP . . . . .	98
5.4.5	com.woty.android.presenter.ModProfileP . . . . .	99
5.4.6	com.woty.android.presenter.PendingQuestsP . . . . .	100
5.4.7	com.woty.android.presenter.SolveQuestP . . . . .	101
5.4.8	com.woty.android.presenter.WorkgroupsRanksP . . . . .	102
5.4.9	com.woty.android.presenter.UsersRanksP . . . . .	103
5.4.10	com.woty.android.presenter.C2dmMessageReceiver . . . . .	104
5.4.11	com.woty.android.presenter.C2dmRegistrationReceiver . . . . .	104
5.5	Specifica componente Util . . . . .	106
5.5.1	com.woty.android.util.JParser . . . . .	106
5.6	Specifica package exceptions . . . . .	107
5.6.1	com.woty.android.exceptions.NotLoggedException . . . . .	107
5.6.2	com.woty.android.exceptions.InvalidArgumentException . . . . .	107
6	Tracciamento Requisiti-Componenti-Classi	109

## Elenco delle tabelle

1	Mappa metodo-funzione per SessionsController . . . . .	41
2	Mappa permessi-metodi . . . . .	41

## Elenco delle figure

1	Esempio esplicativo MVC- User . . . . .	10
2	Classe ApplicationController . . . . .	13
3	Classe ActiveRecord . . . . .	14
4	Classe User . . . . .	15
5	Classe Customer . . . . .	20
6	Classe WorkGroup . . . . .	21
7	Classe Plan . . . . .	22
8	Desktop - Diagramma Classi . . . . .	44
9	Desktop - Classe Manager . . . . .	45
10	Desktop - Classe Exception . . . . .	48
11	Desktop - Classe Client . . . . .	49
12	Desktop - libreria libMaia . . . . .	50
13	Desktop - Classe LoginWindow . . . . .	51
14	Desktop - Classe Account . . . . .	52
15	Desktop - libreria SimpleCrypt . . . . .	54
16	Desktop - Classe Preferences . . . . .	54

17	Mobile - Diagramma Classi . . . . .	57
18	Classe com.woty.android.model.Resource . . . . .	58
19	Classe com.woty.android.model.Configuration . . . . .	59
20	Classe com.woty.android.model.User . . . . .	61
21	Classe com.woty.android.model.Workgroup . . . . .	63
22	Classe com.woty.android.model.Challenge . . . . .	64
23	Classe com.woty.android.model.TrueFalseChallenge . . . . .	66
24	Classe com.woty.android.model.ComboChallenge . . . . .	66
25	Classe com.woty.android.model.MultiOptionChallenge . . . . .	67
26	Classe com.woty.android.model.QRChallenge . . . . .	67
27	Classe com.woty.android.model.Description . . . . .	68
28	Classe com.woty.android.model.TextDescription . . . . .	69
29	Classe com.woty.android.model.ImageDescription . . . . .	69
30	Classe com.woty.android.model.VideoDescription . . . . .	70
31	Classe com.woty.android.model.Quest . . . . .	71
32	Classe com.woty.android.model.QuestHistory . . . . .	72
33	Classe com.woty.android.model.UsersRank . . . . .	73
34	Classe com.woty.android.model.WorkgroupsRank . . . . .	75
35	Classe com.woty.android.model.Session . . . . .	76
36	Classe com.woty.android.view.Vista . . . . .	79
37	Classe com.woty.android.view.Login . . . . .	80
38	Classe com.woty.android.view.Home . . . . .	81
39	Classe com.woty.android.view.ModProfile . . . . .	82
40	Classe com.woty.android.view.PendingQuests . . . . .	83
41	Classe com.woty.android.view.SolveQuest . . . . .	85
42	Classe com.woty.android.view.WorkgroupsRanks . . . . .	87
43	Classe com.woty.android.view.UsersRanks . . . . .	89
44	Classe com.woty.android.view.RankAdapter . . . . .	91
45	Classe com.woty.android.view.PlayerProfile . . . . .	91
46	Classe com.woty.android.view.Help . . . . .	92
47	Classe com.woty.android.view.About . . . . .	94
48	Classe com.woty.android.presenter.Presenter . . . . .	96
49	Classe com.woty.android.presenter.LoginP . . . . .	97
50	Classe com.woty.android.presenter.HomeP . . . . .	98
51	Classe com.woty.android.presenter.PlayerProfileP . . . . .	98
52	Classe com.woty.android.presenter.ModProfileP . . . . .	99
53	Classe com.woty.android.presenter.PendingQuestsP . . . . .	100
54	Classe com.woty.android.presenter.SolveQuestP . . . . .	101
55	Classe com.woty.android.presenter.WorkgroupsRanksP . . . . .	102
56	Classe com.woty.android.presenter.UsersRanksP . . . . .	103
57	Classe com.woty.android.presenter.C2dmMessageReceiver . . . . .	104
58	Classe com.woty.android.presenter.C2dmRegistrationReceiver . . . . .	104
59	Classe com.woty.android.remote.JParser . . . . .	106
60	Classe com.woty.android.exceptions.NotLoggedException . . . . .	107
61	Classe com.woty.android.exceptions.InvalidArgumentException . . . . .	107

## 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<sup>g</sup> 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<sup>g</sup> ) 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\_v4.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*
- Specifica Tecnica:  
vedi documento fornito in allegato *SpecificaTecnica\_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>
- Descrizione Pattern STI:  
<http://martinfowler.com/eaaCatalog/singleTableInheritance.html>
- Documentazione paperclip:  
<https://github.com/thoughtbot/paperclip.git>

## 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\_v3.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\_v4.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\_v4.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\_v4.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\_v4.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<sup>9</sup>. 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 alla 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 verrà descritta l'implementazione di una risorsa secondo il pattern Model-View-Controller, analizzando in dettaglio ogni componente.

Proponiamo un esempio esplicativo della risorsa User.

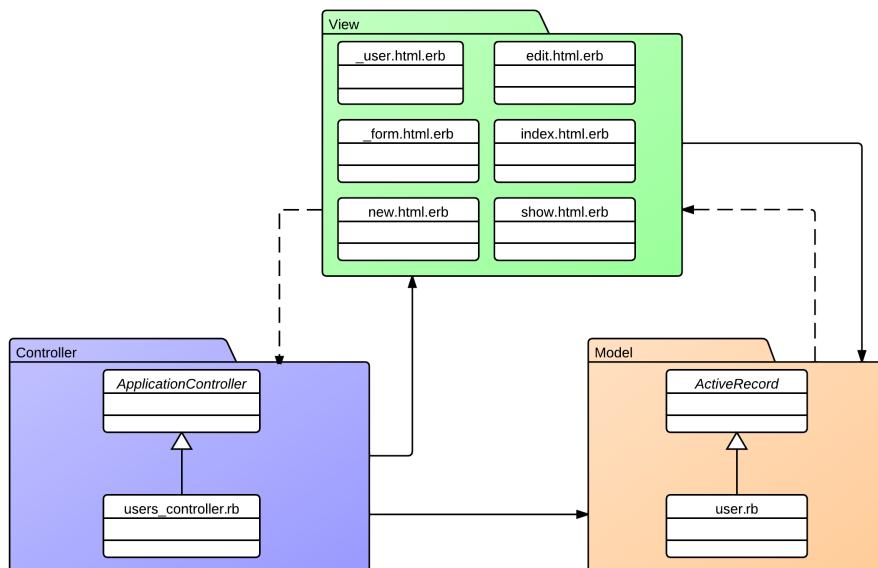


Figura 1: Esempio esplicativo MVC- User

### 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<sup>g</sup>;
- Definisce le validazioni (se esistono) per ognuno dei campi dati;
- Definisce metodi ricorrenti *inerenti alla risorsa* per l'estrazione di dati;
- Definisce eventuali *callback<sup>g</sup>* necessari al funzionamento;

#### Attributi dei modelli

L'implementazione effettiva dell'accesso a campi dati d'istanza dei modelli sul framework<sup>g</sup> utilizzato non fa uso di attributi di istanza del linguaggio Ruby<sup>g</sup> (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". Si ritiene importante l'inclusione dei tali all'interno del documento. Verranno utilizzati i seguenti tag nella descrizione degli attributi per identificare rapidamente le validazioni:

- #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".

Si ritiene importante l'inclusione dei tali all'interno del documento.

### 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<sup>g</sup>* 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<sup>9</sup> 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 (includendo 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<sup>9</sup> necessario alle view della risorsa.

## 3.5 Comportamenti e necessità anomale

Le necessità e i comportamenti anomali saranno descritti specificatamente 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<sup>g</sup>, 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`).

#### Specifiche risorse framework

Di seguito le componenti framework necessarie alle risorse

### 3.5.4 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.

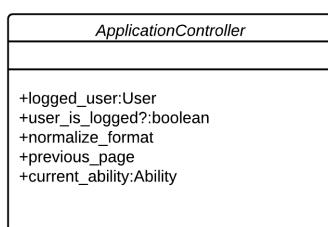


Figura 2: Classe ApplicationController

#### Funzione della classe

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

## Metodi

+ `User logged_user()`

Ritorna l'utente della sessione corrente.

+ `boolean user_is_logged?()`

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

+ `void normalize_format()`

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

+ `void previous_page()`

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

+ `Ability 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.

### 3.5.5 Classe ActiveRecord::Base

La classe rappresenta l'implementazione del design pattern omonimo. Garantisce i metodi basilari per l'esecuzione dei metodi CRUD<sup>9</sup> 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.

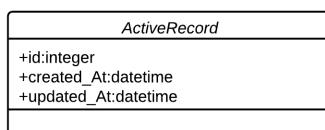


Figura 3: Classe ActiveRecord

## 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.

### 3.6 Specifiche risorse standard

#### 3.6.1 User

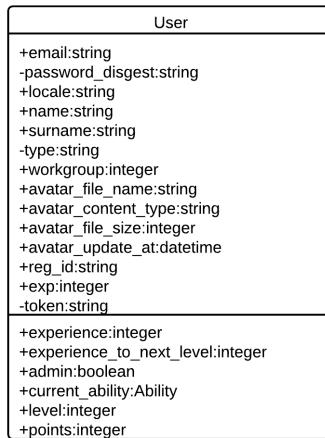


Figura 4: Classe User

#### Funzione della risorsa

User è la risorsa che modella gli utenti registrati al sistema ed è la classe base della gerarchia degli utenti.

La gerarchia è utilizzata per la creazione delle sessioni e l'autorizzazione e rispetta l'implementazione standard descritta precedentemente.

Di seguito verranno elencate le caratteristiche specifiche delle classi coinvolte.

#### Modello

#### Attributi

**+ string email #REQUIRED #UNIQUE**

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

**- string password #REQUIRED**

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

**+ string locale**

Identificativo della lingua dello User.

**+ string name #REQUIRED**

Contiene il nome dell'utente.

**+ string surname #REQUIRED**

Contiene il cognome dell'utente.

**+ string type #REQUIRED**

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

**+ integer workgroup\_id #REQUIRED**

Id del Workgroup<sup>g</sup> cui appartiene l'User.

**+ string avatar**

Avatar dello User

```
+ integer exp
    Punti esperienza dello User

+ string status
    Stato personale dello User
```

## Metodi

```
+ integer experience()
    Ritorna il valore derivante dal calcolo dell'esperienza ottenuta dall'utente.

+ integer experience_to_next_level()
    Ritorna i punti mancanti all'avanzamento di livello.

+ boolean admin }()
    Ritorna true se l'utente corrente è un admin, false altrimenti.

+ string self.subclasses_strings()
    Metodo statico che ritorna un array di stringhe dove ogni stringa identifica una sottoclasse
    di User.

+ Ability current_ability()
    Ritorna un'istanza della classe Ability definita per l'utente connesso.

+ string self.roles()
    Ritorna il nome della classe formattato per l'autorizzazione.

+ integer level()
    Ritorna il livello dell'utente.

+ integer points()
    Ritorna il punteggio dell'utente.

+ integer score_method()
    Ritorna lo score dell'utente e aggiorna il database.
```

## 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
```

## Moduli inclusi

- Achievable::User
- Stats::User

## Controller

Implementazione standard di risorsa.

## View

Implementazione standard di risorsa.

## Decorator

+ void linked\_namesurname()

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

+ void 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.

+ string namesurname()

Ritorna il nome e cognome dell'utente.

+ string avatar()

Ritorna l'html necessario alla visualizzazione dell'avatar.

+ string thumbnail()

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

+ string mini\_thumbnail()

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

+ string level\_bar()

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

+ User self.most\_active()

Ritorna l'utente più attivo in termini sociali.

+ double self.average\_recent\_streams()

Ritorna la media di stream recenti visualizzati dagli utenti.

+ double self.average\_comments()

Ritorna la media dei commenti per utente.

+ double self.average\_likes()

Ritorna la media dei like per utente.

+ double self.average\_level()

Ritorna il livello medio ottenuto dagli utenti.

+ double self.average\_quests()

Ritorna la media delle quest<sup>g</sup> completate per utente.

### 3.6.2 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.

**3.6.3 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.

**3.6.4 MobileUser****Funzione della risorsa**

Specializzazione della classe base User. Implementazione standard.

**Modello****Attributi****+ integer reg\_id**

E' l'id di registrazione a Google C2DM del dispositivo Android<sup>g</sup> associato al Mobile-user<sup>g</sup>. Viene aggiornato ad ogni login dall'applicazione mobile, e settato a nil al logout.

**+ Token token**

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

**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.

**3.6.5 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.

### 3.6.6 Customer

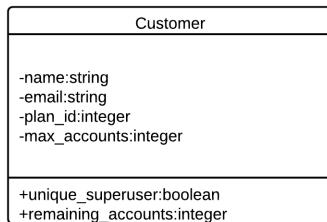


Figura 5: Classe Customer

#### Funzione della risorsa

Risorsa che modella i clienti del fornitore del servizio. Ogni customer ha vari workgroup<sup>9</sup> 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 listate le caratteristiche specifiche delle classi coinvolte.

#### Modello

##### Attributi

- + string email #REQUIRED #UNIQUE  
L'e-mail associata al Customer.
- + string name #REQUIRED #UNIQUE  
Ragione sociale del Customer.
- + integer plan\_id #REQUIRED #UNIQUE  
Id del Plan (tariffa) associato al Customer.
- + integer max\_accounts #REQUIRED  
Numero massimo di licenze disponibili per il Customer.

##### Metodi

- + boolean unique\_superuser()  
Ritorna false se il customer corrente ha più di un superuser, true altrimenti.
- + integer remaining\_accounts()  
Ritorna la differenza tra max\_accounts e gli user attualmente istanziati dal 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

## 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.

### + double self.average\_income

Ritorna la media dei guadagni per cliente.

### + Plan self.preferred\_plan

Ritorna il plan con più preferenze rispetto ai customer.

## 3.6.7 WorkGroup

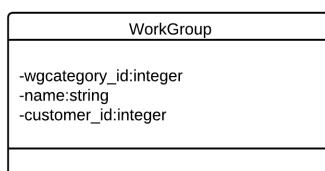


Figura 6: Classe WorkGroup

## Funzione della classe

Classe che rappresenta un reparto di un'azienda cliente, che sono composti dagli utenti facenti parte dello stesso.

Gli utenti in reparti lavorano anche in team, e sono disponibili classifiche di reparto. Sono inoltre utilizzati per l'assegnazione delle quest<sup>g</sup>, in quanto il reparto è una delle caratteristiche per le quali sono influenzate le categorie di rischio.

## Attributi

### - integer wgccategory\_id

Identificativo della categoria di appartenenza del WorkGroup.

### - string name

Nome assegnato al WorkGroup.

### - integer customer\_id

Identificativo dell'azienda/cliente di appartenenza del WorkGroup.

## Metodi

### + integer score()

Ritorna la somma degli score di ogni user presente nel 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.

## 3.6.8 Plan

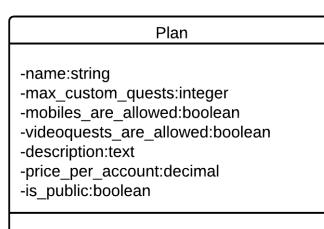


Figura 7: Classe Plan

## Funzione della risorsa

Risorsa che modella le formule d'acquisto del prodotto. Il paradigma *Software as a Service*<sup>9</sup> 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

```
+ string name #REQUIRED #UNIQUE
    Nome del Plan.

+ max_custom_quests #REQUIRED #>0
    Numero massimo di Quest personalizzabili assegnabili ad un Customer che scelga il Plan.

+ boolean videoquests_are_allowed
    Ritorna true se le Quest Video sono disponibili per i Customer, false altrimenti.

+ boolean mobiles_are_allowed
    Ritorna true se il Plan prevede MobileUser, false altrimenti.

+ string description #REQUIRED
    Descrizione testuale del Plan.

+ integer price_per_account #REQUIRED #>=0
    Prezzo per licenza.

+ boolean is_public
    Ritorna 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
```

### Controller

Implementazione standard di risorsa.

### View

Implementazione standard di risorsa.

### Decorator

```
+ public_link
    Ritorna il codice html necessario per un link alla pagina pubblica contenete i piani visibili.

+ active_customer_list
    Ritorna il codice html necessario per la lista di sottoscrittori del piano, linkati.

+ string self.statistics
    Ritorna una stringa indicante il numero di piani visibili pubblicamente.
```

### 3.6.9 Quest

#### Funzione della classe

La classe modella le quest<sup>g</sup> 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.

## Attributi

- + `integer granted_exp`  
Rappresenta l'esperienza guadagnata nel caso di corretta risoluzione della Quest.
- + `integer quest_challenge_id`  
Identificativo della sezione QuestChallenge che compone la Quest.
- + `integer quest_description_id`  
Identificativo della sezione QuestDescription che compone la Quest.

## Metodi

- + `boolean correct?(ans)`  
Ritorna `true` se la quest è corretta per la risposta passata con il parametro.
- + `void self.periodical_assign()`  
Crea e assegna periodicamente una quest a determinati User. Le politiche di assegnazione sono le seguenti:
  - la categoria di rischio associata alla quest è associata al workgroup dove lo user lavora;
  - la quest non è attualmente assegnata all'utente;
  - lo user ha al massimo 5 quest già assegnate ancora da svolgere.
- + `string quest_challenge_type()`  
Ritorna il tipo della sfida associata alla quest.
- + `string quest_description_type()`  
Ritorna il tipo della descrizione associata alla quest.
- + `string self.subclasses_strings()`  
Ritorna un vettore di stringhe corrispondenti ai nomi delle sottoclassi.
- + `void self.c2dmNotify(User)`  
Notifica ad un Mobile User l'assegnazione di una quest tramite c2dm.

## 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`

## 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.

### 3.6.10 QuestDescription

#### Funzione della risorsa

Classe base della gerarchia inherente le tipologie di descrizione di una quest. La gerarchia è utilizzata per la composizione di una quest.

Rispetta l'implementazione standard per gerarchie descritte precedentemente.  
Di seguito verranno elencate le caratteristiche specifiche delle classi coinvolte.

#### Modello

#### Attributi

+ `string name #REQUIRED`

Specifica il nome relativo alla QuestDescription.

#### Associazioni

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

- `has_many quests`

#### Controller

Implementazione standard di risorsa.

#### View

Implementazione standard di risorsa.

### 3.6.11 TextDescription

#### Funzione della risorsa

La classe modella una QuestDescription che contiene un testo.

#### Modello

#### Attributi

+ `string text #REQUIRED`

Contiene il testo da visualizzare.

#### Associazioni

Solamente quelle ereditate da QuestDescription.

#### Controller

Implementazione standard di risorsa.

#### View

Implementazione standard di risorsa.

### 3.6.12 ImageDescription

#### Funzione della risorsa

La classe modella una QuestDescription che contiene un'immagine.

#### Modello

#### Attributi

+ `Image img #REQUIRED`

Contiene l'immagine da visualizzare, la persistenza delle immagini è stata implementata con la gemma paperclip, per la quale si faccia riferimento alla sezione 1.4.2.

#### Associazioni

Solamente quelle ereditate da QuestDescription.

#### Controller

Implementazione standard di risorsa.

#### View

Implementazione standard di risorsa.

### 3.6.13 VideoDescription

#### Funzione della risorsa

La classe modella una QuestDescription che contiene un video.

#### Modello

#### Attributi

+ `string video #REQUIRED #UNIQUE`

Contiene l'indirizzo del video da visualizzare, il quale sarà stato precedentemente caricato su Youtube, e la sua visualizzazione nel browser avverrà tramite una funzione embed.

#### Associazioni

Solo quelle ereditate da QuestDescription.

#### Controller

Implementazione standard di risorsa.

#### View

Implementazione standard di risorsa.

### 3.6.14 QuestChallenge

#### Funzione della classe

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

+ `string question #REQUIRED`

Contiene il quesito rivolto all'utente sotto forma di stringa.

- `string type #REQUIRED`

Specializza il tipo di QuestChallenge. Necessario per il triggering del pattern STI, il quale è trattato nei riferimenti informativi, sezione 1.4.2.

+ `string name #REQUIRED #UNIQUE`

Nome relativo alla QuestChallenge.

## Metodi

+ `void tfstatements_must_not_have_empty_values()`

Controlla che l'hash tfstatements non abbia statement nulli, altrimenti solleva un errore.

+ `void tfstatements_must_be_boolean()`

Controlla che per ogni statement dell'hash tfstatement ci sia un valore booleano associato.

## Associazioni

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

- `has_many quests`

## Controller

Implementazione standard di risorsa.

+ `sanitize_tfstatements`

Metodo di controllo.

## View

Implementazione standard di risorsa.

### 3.6.15 ComboChallenge

#### Funzione della classe

Classe che rappresenta un quesito a risposta multipla con una ed una sola risposta corretta.

## Modello

### Attributi

+ `integer rstatement #REQUIRED #>0`

Identifica la risposta corretta.

- `text tfstatements #REQUIRED`

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

## Metodi

+ `boolean rstatement_must_be_valid()`

Controlla se il campo `rstatement` ha un valore legale.

+ `boolean correct(String)`

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

+ `integer tfNumber()`

Restituisce il numero di quesiti della combo challenge.

## Associazioni

Solo quelle ereditate da `QuestChallenge`.

## Controller

Implementazione standard di risorsa.

## View

Implementazione standard di risorsa.

### 3.6.16 TrueFalseChallenge

## Associazioni

Solo quelle ereditate da `QuestChallenge`.

## Controller

Implementazione standard di risorsa.

## View

Implementazione standard di risorsa.

### 3.6.17 TrueFalseChallenge

## Funzione della risorsa

La classe modella una tipologia particolare di `QuestChallenge` contenente più affermazioni, ognuna delle quale può essere vera o falsa.

## Modello

## Attributi

+ `text tfstatements #REQUIRED`

Contiene la serializzazione YAML dell'hash relativo alle `n` affermazioni e ai rispettivi valori.

## Metodi

+ `boolean correct?(ans)`

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

## Associazioni

Solo quelle ereditate da `QuestChallenge`

## Controller

Implementazione standard di risorsa.

## View

Implementazione standard di risorsa.

### 3.6.18 MultiOptionChallenge

#### Funzione della risorsa

La classe modella una tipologia particolare di `QuestChallenge` contenente più affermazioni, ognuna delle quale 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

+ `text tfstatements #REQUIRED`

Contiene la serializzazione YAML dell'hash relativo alle `n` affermazioni e ai rispettivi valori.

##### Metodi

+ `boolean correct?(ans)`

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

#### Associazioni

Solo quelle ereditate da `QuestChallenge`.

## Controller

Implementazione standard di risorsa.

## View

Implementazione standard di risorsa.

### 3.6.19 Achievement

#### Funzione della risorsa

Gli achievement<sup>g</sup> 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

+ `string name #REQUIRED #UNIQUE`

Nome identificativo dell'achievement.

- `text description #REQUIRED`

Descrizione dell'achievement.

## Metodi

+ `check_and_award(user)`

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

+ `boolean valid_for?(user)`

Ritorna `true` se l'achievement è completato per lo user, `false` altrimenti.

## 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`

## 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.

## 3.6.20 Like

### Funzione della risorsa

Risorsa che modella i like effettuati dagli utenti. Attualmente è possibile effettuare i like solo su elementi membri della gerarchia `StreamElement`.

### Modello

#### Attributi

+ `integer user_id #REQUIRED`

Lo user che effettua il like.

- `integer 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

## Controller

Implementazione standard di risorsa.

## View

Implementazione standard di risorsa.

### 3.6.21 Comment

#### Funzione della risorsa

Risorsa che modella i commenti effettuati dagli utenti. Ad oggi è possibile commentare solamente elementi membri della gerarchia StreamElement.

#### Modello

##### Attributi

- + integer user\_id #REQUIRED  
Utente che effettua il commento.
- integer stream\_element\_id #REQUIRED  
StreamElement commentato.
- + text body #REQUIRED  
Corpo del commento.

##### Metodi

- + boolean valid\_user\_and\_element()  
Controlla se lo user che esegue l'azione è valido.

## 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

## Controller

Implementazione standard di risorsa.

## View

Implementazione standard di risorsa.

### 3.6.22 StreamElement

#### Funzione della risorsa

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

+ string type #REQUIRED #UNIQUE  
Tipo di StreamElement per "Single Table Inheritance".

##### Metodi

+ void self.subklasses()  
Ritorna un array di oggetti di tipo class contenente le sottoclassi.  
  
+ UserDecorator owner()  
Ritorna un’istanza di UserDecorator che decora il proprietario dello StreamElement.  
  
+ string title()  
Ritorna un titolo per lo StreamElement.  
  
+ text description()  
Ritorna una descrizione per lo StreamElement.

#### 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

#### 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.

### 3.6.23 Specifica Classe QuestHistory

#### Funzione della risorsa

Classe che 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

+ integer quest\_id #REQUIRED

Identifica la Quest relativa alla QuestHistory.

+ integer user\_id #REQUIRED

Identifica lo User relativo alla QuestHistory.

- boolean 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).

- boolean successful

`true` se la Quest è stata affrontata correttamente, `false` altrimenti.

### Metodi

+ void 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.

+ string quest\_name()

Ritorna il nome della Quest relativa alla QuestHistory.

+ integer quest\_exp()

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

### Associazioni

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

- belongs\_to quest

- belongs\_to user

### Controller

Implementazione standard di risorsa.

### View

Implementazione standard di risorsa.

### 3.6.24 Ticket

#### Funzione della risorsa

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

## Modello

### Attributi

- Derivati da ActiveRecord.
- `string title`  
Oggetto del Ticket.
- `integer customer_id`  
Identifica il Customer relativo al SuperUser che crea il Ticket.
- `text 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.

## 3.6.25 CompletedAchievement

### Funzione della risorsa

Rappresenta un achievement ottenuto da un utente.

## Modello

### Attributi

- Derivati da ActiveRecord.
- + `integer user_id`  
Id dell'utente che ha ottenuto l'achievement.
- + `integer achievement_id`  
Id dell'achievement ottenuto.

### Metodi

- + `boolean stop_duplicates()`  
False se l'user ha già ottenuto l'achievement, true altrimenti.
- + `CompletedAchievementElement create_and_link_completed_achievement_element()`  
Crea un CompletedAchievementElement associato all'istanza.
- + `boolean valid_user_and_achievement()`  
Controlla che lo user a cui viene associato il CompletedAchievement sia valido.

## Associazioni

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

- belongs\_to achievement
- belongs\_to user

## Controller

Implementazione standard di risorsa.

## View

Implementazione standard di risorsa.

### 3.6.26 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.
- + integer completed\_achievement\_element\_id #REQUIRED  
Id del corrispondente CompletedAchievementElement.

#### Metodi

- + string title()  
Specializzazione dalla classe base
- + text description()  
Specializzazione dalla classe base

## 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

## Controller

Implementazione standard di risorsa.

## View

Implementazione standard di risorsa.

### 3.6.27 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.

#### Metodi

- + `boolean 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.
- + `boolean existing_achievement()`  
Controlla che l'achievement associato alla condition esista effettivamente.

#### Associazioni

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

- `belongs_to achievement`

#### Controller

Implementazione standard di risorsa.

#### View

Implementazione standard di risorsa.

### 3.6.28 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`.

#### Attributi

- + `integer workgroup_id`  
Identifica il `Workgroup` al quale la `CustomQuest` è associata.

## Associazioni

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

- belongs\_to :workgroup

## Controller

Implementazione standard di risorsa.

## View

Implementazione standard di risorsa.

### 3.6.29 QuestAssociation

#### Funzione della classe

Classe che relaziona le QuestCategory con i Workgroup.

#### Attributi

+ integer workgroup\_id

Identifica il Workgroup al quale la QuestAssociation è relazionata.

+ integer quest\_category\_id

Identifica la QuestCategory al quale la QuestAssociation è relazionata.

## Associazioni

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

- belongs\_to :workgroup

- belongs\_to :quest\_category

## Controller

Implementazione standard di risorsa.

## View

Implementazione standard di risorsa.

### 3.6.30 QuestCategory

#### Funzione della classe

Identifica una categoria di rischio al quale una SharedQuest è associata.

#### Attributi

+ string name

Il nome testuale della QuestCategory

## Associazioni

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

- has\_many :quests
- has\_many :quest\_associations
- has\_many :workgroups, :through => :quest\_associations

### 3.6.31 SharedQuest

#### Funzione della classe

Rappresenta una Quest condivisa tra più Workgroup.

#### Attributi

- + integer quest\_category\_id  
Identifica la QuestCategory associata.

## Associazioni

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

- belongs\_to :quest\_category
- has\_many :workgroups, :through => :quest\_associations

### 3.6.32 SignupElement

#### Funzione della risorsa

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

#### Modello

#### Attributi

Solo quelli ereditati da StreamElement

#### Metodi

- + string title()  
Specializzazione dalla classe base.
- + text description()  
Specializzazione dalla classe base.

## Associazioni

Solo quelle ereditate da StreamElement

#### Controller

Implementazione standard di risorsa.

#### View

Implementazione standard di risorsa.

### 3.6.33 StatusChangeElement

#### Funzione della risorsa

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

#### Modello

#### Attributi

- Derivati da ActiveRecord.

+ string new\_status #REQUIRED

Nuovo stato dell'utente.

#### Metodi

+ string title()

Specializzazione dalla classe base.

+ text description()

Specializzazione dalla classe base.

#### Associazioni

Solo quelle ereditate da StreamElement.

#### Controller

Implementazione standard di risorsa.

#### View

Implementazione standard di risorsa.

### 3.6.34 Wgcategory

#### Funzione della classe

Classe che rappresenta una tipologia di Workgroup.

#### Relazioni con altre classi

Derivata da ActiveRecord, è relazionata con Workgroup.

#### Attributi

+ string name

Il nome della tipologia di Workgroup.

+ text description

Descrizione testuale.

#### Associazioni

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

- has\_many :workgroups

### 3.7 Altre risorse

#### 3.7.1 XmlrpcController

Il controller XmlrpcController, descritto nel file `app/controllers/xmlrpc_controller.rb`, contiene i metodi accessibili tramite richiesta XMLRPC. E' stato deciso di usare questo sistema per tutte le richieste difficilmente mappabili come RESTful<sup>9</sup>. 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 coppia 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.

#### 3.7.2 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

###### + `string email #REQUIRED`

Indirizzo email.

###### + `string password #REQUIRED`

Password.

###### + `boolean id`

Id utente.

## Metodi

**+ void initialize(attributes)**

Costruisce l'oggetto con gli attributi parametro.

**+ boolean 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
destroy	Distruzione di una sessione
edit, update, index, show	Nessuna

## Metodi

**+ index**

Renderizza una risposta nulla (http 200 ok, body vuoto).

**+ new**

Renderizza una pagina di login.

**+ show**

Renderizza una risposta nulla (http 200 ok, body vuoto).

**+ edit**

Renderizza una risposta nulla (http 200 ok, body vuoto).

**+ create**

Crea una sessione attiva se i parametri sono corretti.

**+ update**

Renderizza una risposta nulla (http 200 ok, body vuoto).

**+ destroy**

Distrugge la sessione attiva corrente se esiste.

**View** Solamente \_\_form.html.erb, index.html.erb, new.html.erb come implementazione standard.

### 3.7.3 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

*(Continua alla pagina successiva)*

(Continua dalla pagina precedente)

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 user

L'utente per il quale si vogliono controllare i permessi.

## Metodi

### + void 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.

### + void guests()

Definisce i permessi per gli utenti non registrati.

### + void desktop\_user()

Definisce i permessi degli utenti desktop.

### + void mobile\_users()

Definisce i permessi degli utenti mobile.

### + void superusers()

Definisce i permessi dei superuser.

### + void admins()

Definisce i permessi degli admin.

## 3.7.4 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.

## Metodi

### + void home()

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

**+ void 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.

**+ void 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.

**+ void denied()**

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

**+ void 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.

### 3.7.5 Moduli

#### Achievable::User

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

**+ integer number\_of\_comments()**

Ritorna il numero di commenti effettuati da un utente.

**+ integer number\_of\_status\_changes()**

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

#### Requirements

Ogni metodo del modulo rappresenta un requisito di sistema che deve essere validato.

**+ boolean self.there\_should\_be\_an\_admin()**

True se esiste almeno un admin.

**+ boolean self.there\_should\_be\_a\_customer()**

True se esiste almeno un customer.

**+ boolean self.there\_should\_be\_a\_plan()**

True se esiste almeno un plan.

#### Stats::User

Ogni metodo definisce dei valori notevoli a fini statistici per gli user.

**+ integer number\_of\_recent\_streams()**

Numero di stream recenti visualizzabili dallo user

**+ integer 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<sup>9</sup>, 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 Classi

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

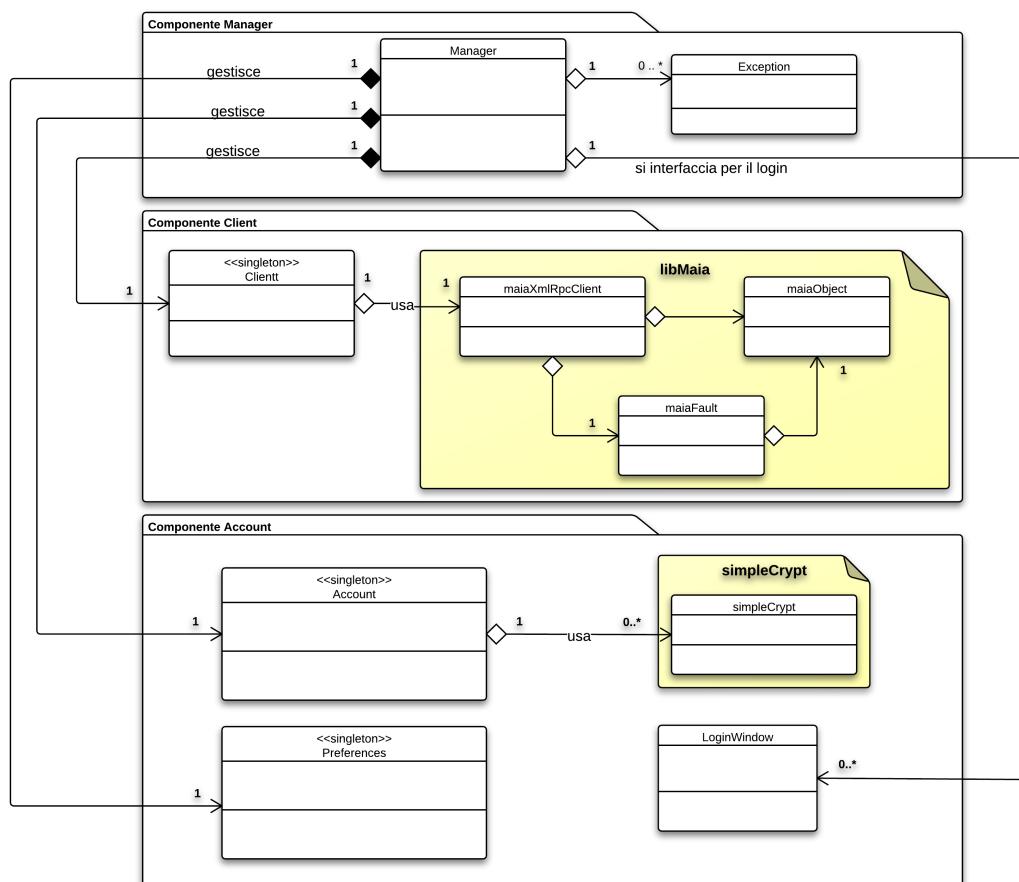


Figura 8: Desktop - Diagramma Classi

### 4.2 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.

#### 4.2.1 Specifica classe Manager

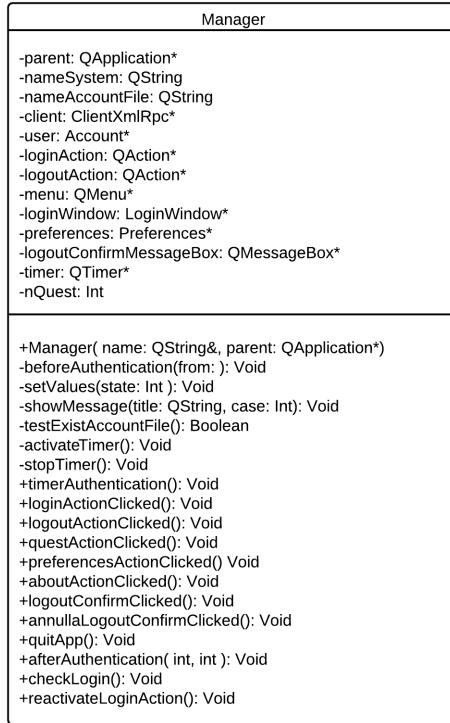


Figura 9: Desktop - Classe Manager

#### 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'unica istanza della classe Client che gestisce le comunicazioni con la piattaforma Woty.

**- Account\* user**

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

**- Preferences\* preferences**

Puntatore all'unica istanza 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 Icon<sup>9</sup> 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'unica istanza 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 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.

#### 4.2.2 Specifica classe Exception

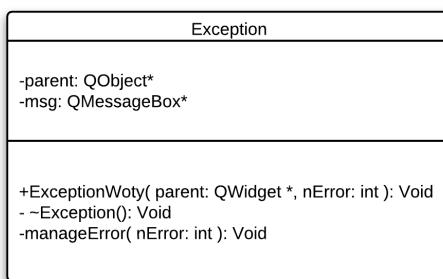


Figura 10: Desktop - Classe Exception

#### 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.

## 4.3 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.

### 4.3.1 Specifica classe Client

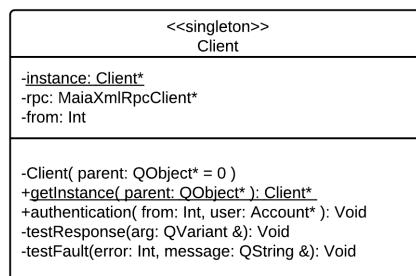


Figura 11: Desktop - Classe Client

### 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 contenente 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.

### 4.3.2 Specifica libreria LibMaia

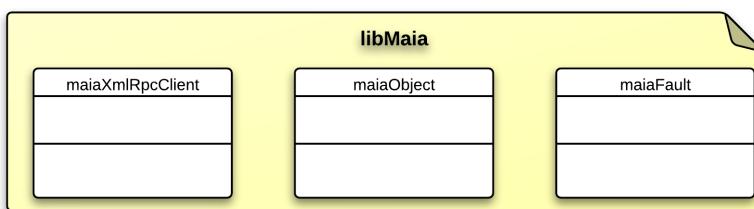


Figura 12: Desktop - libreria libMaia

### 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`

## 4.4 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.

### 4.4.1 Specifica classe LoginWindow

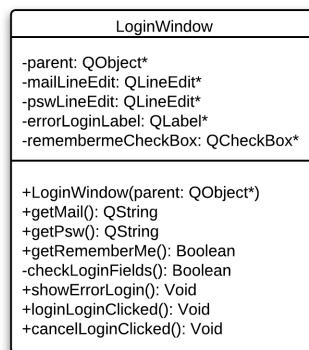


Figura 13: Desktop - Classe LoginWindow

#### 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.

### 4.4.2 Specifica classe Account

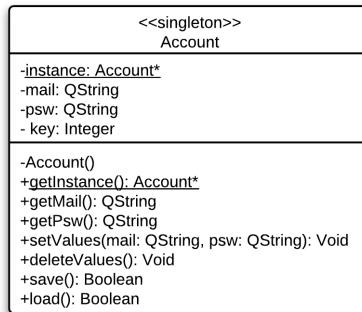


Figura 14: Desktop - Classe Account

### 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.

#### 4.4.3 Specifica libreria simpleCrypt

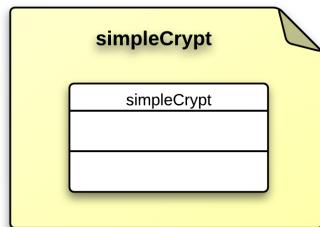


Figura 15: Desktop - libreria SimpleCrypt

#### 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

#### 4.4.4 Specifica classe Preferences

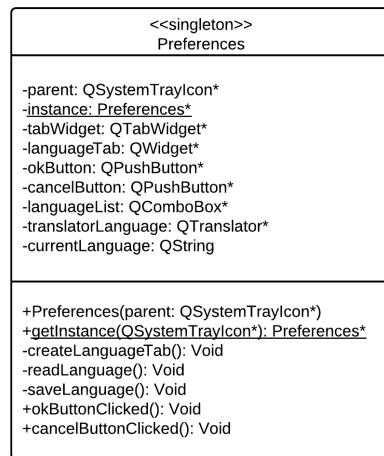


Figura 16: Desktop - Classe Preferences

## 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<sup>g</sup> utilizzatori di un dispositivo Android<sup>g</sup>. 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 delle Classi

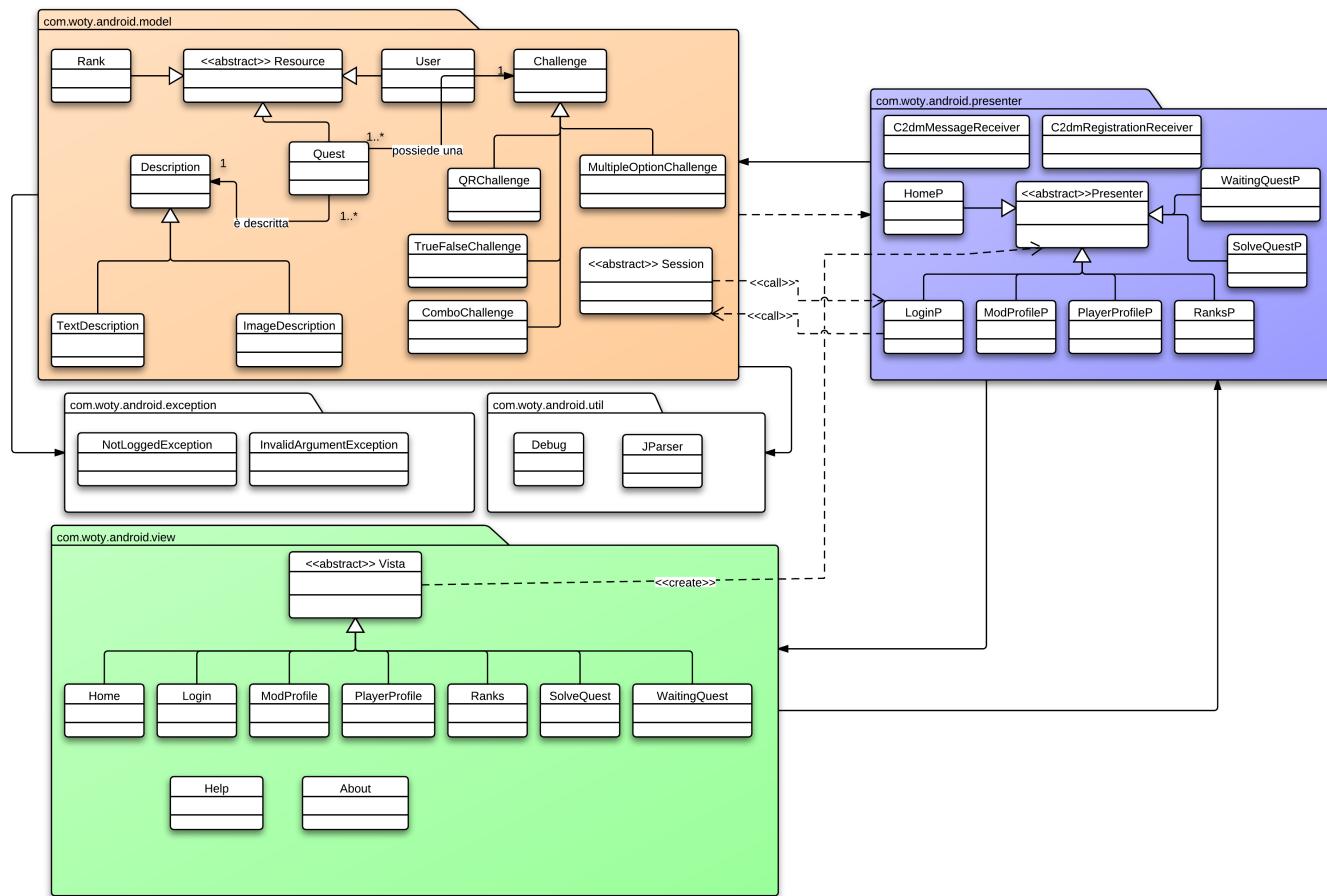


Figura 17: Mobile - Diagramma Classi

## 5.2 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.

### 5.2.1 com.woty.android.model.Resource

Resource
<pre>-callMethod(method: String, arguments: Object[]): Object #getResource(resoucePath: String, resourceId: String, params: String[]): String #setResource(resourcePath: String, resourceId: String, params: String[]): Boolean #newResource(resoucePath: String, resourceId: String, params: String[]): Boolean +getImage(imageFolder: String, imageSection: String, resourceId: Int, imageFileName: String): Bitmap</pre>

Figura 18: 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.Workgroup
  - com.woty.android.model.Challenge
  - com.woty.android.model.Description
  - com.woty.android.model.Quest
  - com.woty.android.model.QuestHistory
  - com.woty.android.model.WorkgroupRank
  - com.woty.android.model.UserRank
- Utilizza le classi:
  - org.xmlrpc.android.XMLRPCClient
  - org.apache.http.client.HttpClient

#### Metodi

`~ static final Object callMethod(String method, Object[] arguments)`  
 Metodo utilizzato per invocare un generico metodo xml-rpc.

`# static final String getResource(String resoucePath, 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.
```

### 5.2.2 com.woty.android.model.Configuration

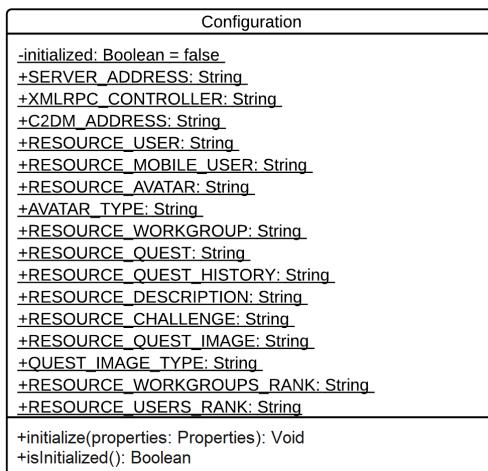


Figura 19: Classe com.woty.android.model.Configuration

#### Funzione della classe

Classe contenente gli indirizzi necessari alle classi di modello per le comunicazioni con il server. Si collega ad un file xml esterno **config.xml** con tipo **properties** tramite l'utilizzo di **java.util.Properties**.

#### Relazioni con altre classi

- È utilizzata da:
  - com.woty.android.model.User
  - com.woty.android.model.Workgroup
  - com.woty.android.model.Challenge
  - com.woty.android.model.Description
  - com.woty.android.model.Quest
  - com.woty.android.model.QuestHistory
  - com.woty.android.model.WorkgroupRank
  - com.woty.android.model.UserRank
- Utilizza le classi:
  - java.util.Properties
  - com.woty.android.exceptions.InvalidArgumentException

## Attributi

- static boolean initialized = false  
Flag che indica se l'oggetto è stato inizializzato a partire dal file config.
- + static String SERVER\_ADDRESS  
Contiene l'indirizzo del server.
- + static String XMLRPC\_CONTROLLER  
Contiene l'indirizzo al controller xml-rpc sul server.
- + static String C2DM\_ADDRESS  
Contiene l'indirizzo e-mail collegato al servizio C2DM per le notifiche push.
- + static String RESOURCE\_USER  
Contiene l'indirizzo della cartella degli user sul server.
- + static String RESOURCE\_MOBILE\_USER  
Contiene l'indirizzo della cartella dei mobile-user sul server.
- + static String RESOURCE\_AVATAR  
Contiene l'indirizzo della cartella degli avatar degli utenti nel server.
- + static String AVATAR\_TYPE  
Contiene il nome della cartella sul server degli avatar con grandezza dedicata ai dispositivi mobile.
- + static String RESOURCE\_WORKGROUP  
Contiene l'indirizzo della cartella dei workgroup sul server.
- + static String RESOURCE\_QUEST  
Contiene l'indirizzo della cartella delle quest sul server.
- + static String RESOURCE\_QUEST\_HISTORY  
Contiene l'indirizzo della cartella "QuestHistory" sul server.
- + static String RESOURCE\_DESCRIPTION  
Contiene l'indirizzo della cartella delle descrizioni sul server.
- + static String RESOURCE\_CHALLENGE  
Contiene l'indirizzo della cartella delle sfide sul server.
- + static String RESOURCE\_QUEST\_IMAGE  
Contiene l'indirizzo della cartella delle immagini dedicate alle quest sul server.
- + static String QUEST\_IMAGE\_TYPE  
Contiene il nome della cartella sul server dedicata alla tipo delle immagini da visualizzare sui dispositivi mobile.
- + static String RESOURCE\_WORKGROUPS\_RANK  
Contiene l'indirizzo della cartella delle classifiche dei workgroup.
- + static String RESOURCE\_USERS\_RANK  
Contiene l'indirizzo della cartella delle classifiche degli user.

## Metodi

- + static void initialize(Properties properties) throws InvalidArgumentException  
Metodo che carica le stringhe contenute nel file config.xml nelle variabili della classe.  
Inoltre imposta il flag initialized a true.
- + static boolean isInitialized()  
Metodo che ritorna il valore del flag initialized a true.

### 5.2.3 com.woty.android.model.User

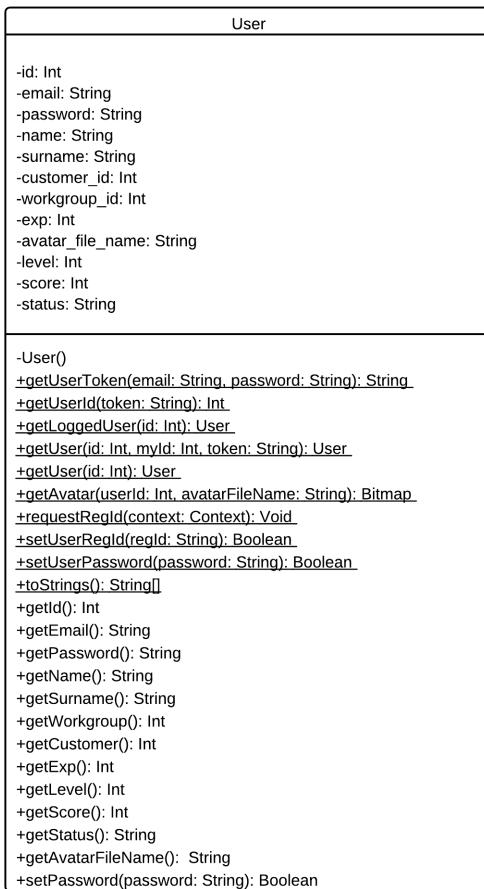


Figura 20: Classe com.woty.android.model.User

#### 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.model.Configuration`
  - `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 customer\_id**  
Identificativo dell'azienda dello user.
- **int workgroup\_id**  
Numero identificativo del workgroup a cui l'utente è associato.
- **int exp**  
Indica l'esperienza dell'utente.
- **int score**  
Indica il punteggio dell'utente.
- **int level**  
Indica il livello dell'utente.
- **String status**  
Stato dell'utente.
- **String avatar\_file\_name**  
Nome dell'immagine usata dall'utente come 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 setUserPassword(String password) throws NotLoggedException
    Metodo che modifica la password dello User passando identificativo, token dello stesso e
    la nuova password.

+ String[] toStrings()
    Metodo che ritorna un'array di stringhe rappresentante l'oggetto.
```

#### 5.2.4 com.woty.android.model.Workgroup

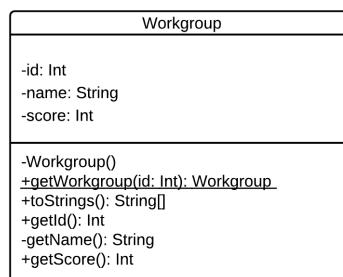


Figura 21: Classe com.woty.android.model.Workgroup

#### Funzione della classe

Classe che rappresenta un generico workgroup 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.model.Configuration`
  - `com.woty.android.model.Session`
  - `com.woty.android.util.JParser`
  - `com.woty.android.exceptions.NotLoggedException`

#### Attributi

- `int id`  
Identificativo del workgroup.
- `String name`  
Contiene il nome del workgroup.
- `int score`  
Contiene il punteggio complessivo del workgroup.

## Metodi

### - `Workgroup()`

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

### + `static final Workgroup getWorkgroup(int id) throws NotLoggedException`

Metodo che ottiene un workgroup dal server fornendo il relativo id.

### + `String[] toStrings()`

Metodo che ritorna un'array di stringhe rappresentante l'oggetto.

### + `int getId()`

Metodo *getter* dell'attributo `id`.

### + `String getName()`

Metodo *getter* dell'attributo `name`.

### + `int getScore()`

Metodo *getter* dell'attributo `score`.

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 *setter* per la password dell'utente.

### + `void setAvatar(String s)`

Metodo *setter* per l'avatar dell'utente.

### + `void setExp(int _exp)`

Metodo *setter* per l'esperienza dell'utente.

### + `void setScore(int _score)`

Metodo *setter* per il punteggio dell'utente.

### + `void setLevel(int _level)`

Metodo *setter* per il livello dell'utente.

### + `void setStatus(String _status)`

Metodo *setter* per lo stato dell'utente.

## 5.2.5 com.woty.android.model.Challenge

<i>Challenge</i>
<pre>#id: Int #name: String #question: String #rtfstatements: HashMap&lt;String, String&gt;  #Challenge() +getChallenge(challengeld: Int, type: String): Challenge +toStrings(): String[] +getId(): Int +getName(): String +getQuestion(): String +getRtfstatements(): HashMap&lt;String, String&gt;</pre>

Figura 22: 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.model.Configuration`
  - `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.
```

### 5.2.6 com.woty.android.model.TrueFalseChallenge

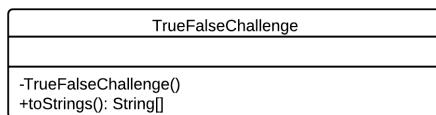


Figura 23: Classe com.woty.android.model.TrueFalseChallenge

#### 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.

### 5.2.7 com.woty.android.model.ComboChallenge

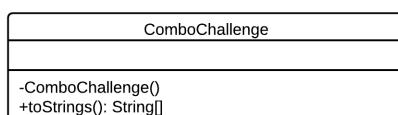


Figura 24: Classe com.woty.android.model.ComboChallenge

#### 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.

### 5.2.8 com.woty.android.model.MultiOptionChallenge

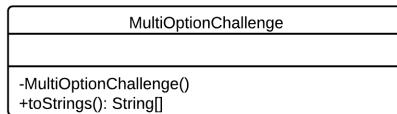


Figura 25: Classe com.woty.android.model.MultiOptionChallenge

#### 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.

### 5.2.9 com.woty.android.model.QRChallenge

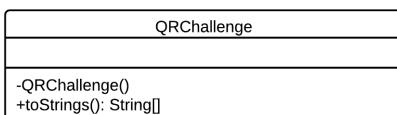


Figura 26: Classe com.woty.android.model.QRChallenge

#### Funzione della classe

Classe astratta che rappresenta una sfida con compito cognitivo che includa la scansione di QR-Code<sup>g</sup> 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.

### 5.2.10 com.woty.android.model.Description

<i>Description</i>
#id: Int ">#name: String
#Description() +getDescription(descriptionId: Int, type: String): Description +toStrings(): String[] +getId(): Int +getName(): String

Figura 27: 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`
  - `com.woty.android.model.VideoDescription`
- Utilizza le classi:
  - `com.woty.android.model.Configuration`
  - `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`.

### 5.2.11 com.woty.android.model.TextDescription

TextDescription
-text: String
-TextDescription()
+toStrings(): String[]
+getText(): String

Figura 28: 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**.

### 5.2.12 com.woty.android.model.ImageDescription

ImageDescription
-img_file_name: String
-ImageDescription()
+toStrings(): String[]
+getDescriptionImage(imageDescriptionId: Int, String imageName): Bitmap
+getDescriptionImage(): Bitmap
+getImageName(): String

Figura 29: 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.

## 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 imageFileName)`  
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`.

## 5.2.13 com.woty.android.model.VideoDescription

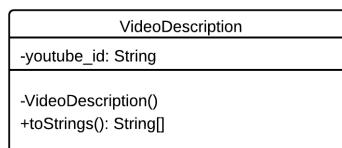


Figura 30: Classe com.woty.android.model.VideoDescription

## Funzione della classe

Classe che rappresenta una descrizione con video 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 youtube_id`  
Stringa contenente l'URL del video.

## Metodi

- `VideoDescription()`  
Viene definito il costruttore privato in quanto una `VideoDescription` è istanziabile solamente in ritorno da una richiesta al server.
- + `String[] toStrings()`  
Metodo ridefinito dalla superclasse che ritorna una stringa rappresentante l'oggetto.

### 5.2.14 com.woty.android.model.Quest

Quest
<pre>-id: Int -name: String -granted_exp: Int -quest_description_id: Int -quest_challenge_id: Int -quest_description_type: String -quest_challenge_type: String -description: Description = null -challenge: Challenge = null</pre>
<pre>-Quest() +getQuest(questId: Int): Quest +submit(check: Int[], questId: Int): Boolean_ +getId(): Int +getName(): String +getGrantedExp(): Int +getDescription(): Description +getChallenge(): Challenge +getChallengeType(): String +getDescriptionType(): String</pre>

Figura 31: 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.Configuration`
  - `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.
- **`int 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.

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

Metodo che invia al server la risposta alla quest effettuata da uno User.

+ **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.

## 5.2.15 com.woty.android.model.QuestHistory

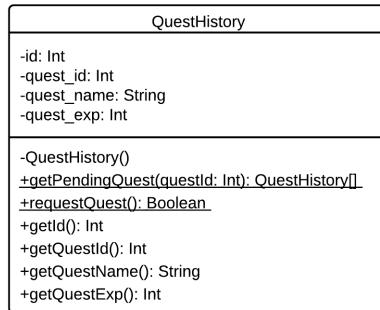


Figura 32: Classe com.woty.android.model.QuestHistory

## Funzione della classe

Classe che rappresenta una questa 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.Configuration`
  - `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.
- + `static final boolean requestQuest() throws NotLoggedException`  
Metodo che richiede l'assegnamento di una quest 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`.

### 5.2.16 com.woty.android.model.UsersRank

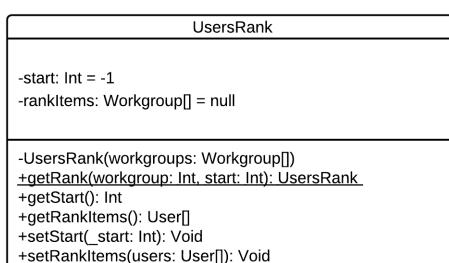


Figura 33: Classe com.woty.android.model.UsersRank

## Funzione della classe

Classe che rappresenta una classifica a livello di utenti 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.Configuration`
  - `com.woty.android.model.User`
  - `com.woty.android.model.Session`
  - `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

- `int start = -1`  
Posizione di inizio da visualizzare nella classifica.
- `User[] rankItems = null`  
Lista degli utenti nella classifica.

## Metodi

- `UsersRank()`  
Viene definito il costruttore privato in quanto una classifica di utenti è istanziabile solamente in ritorno dalla richiesta al server.
- + `static final UsersRank getRank(int workgroup, int start) throws NotLoggedException`  
Metodo che ottiene un UsersRank, che consiste in un array di User , dal server fornendo identificativo del workgroup e posizione di inizio della classifica.
- + `int getStart()`  
Metodo *getter* dell'attributo `start`.
- `User[] getRankItems()`  
Metodo *getter* dell'attributo `rankItems`.
- + `setStart(int start)`  
Metodo *setter* dell'attributo `start`.
- `User[] setRankItems(User[] users)`  
Metodo *setter* dell'attributo `rankItems`.

### 5.2.17 com.woty.android.model.WorkgroupsRank

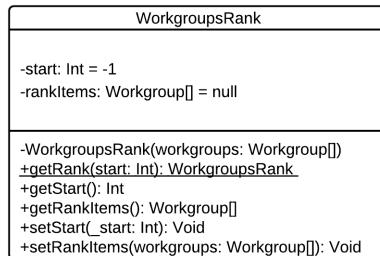


Figura 34: Classe com.woty.android.model.WorkgroupsRank

#### Funzione della classe

Classe che rappresenta una classifica a livello di workgroup 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.Configuration`
  - `com.woty.android.model.Workgroup`
  - `com.woty.android.model.Session`
  - `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

- `int start = -1`  
Posizione di inizio da visualizzare nella classifica.
- `Workgroup[] rankItems = null`  
Lista dei workgroup nella classifica.

#### Metodi

- `WorkgroupsRank(Workgroup[] workgroups)`  
Viene definito il costruttore privato in quanto una classifica di utenti è istanziabile solamente in ritorno dalla richiesta al server.
- + `static final WorkgroupsRank getRank(int start) throws NotLoggedException`  
Metodo che ottiene un WorkgroupsRank, che consiste in un array di Workgroup , dal server fornendo identificativo dello user che ha effettuato la richiesta e posizione di inizio della classifica.
- + `int getStart()`  
Metodo getter dell'attributo `start`.

`User[] getRankItems()`  
 Metodo *getter* dell'attributo `rankItems`.

`+ setStart(int start)`  
 Metodo *setter* dell'attributo `start`.

`User[] setRankItems(User[] users)`  
 Metodo *setter* dell'attributo `rankItems`.

#### 5.2.18 com.woty.android.model.Session



Figura 35: 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.WorkgroupsRanksP`
  - `com.woty.android.presenter.UsersRanksP`

- com.woty.android.presenter.PlayerProfileP
- com.woty.android.presenter.PendingQuestP
- 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 updateLoggedUser(User u)
    Metodo che aggiorna l'utente autenticato.
```

## 5.3 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.

### 5.3.1 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 36: 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.UsersRanks`
  - `com.woty.android.view.WorkgroupsRanks`
  - `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.

### 5.3.2 com.woty.android.view.Login

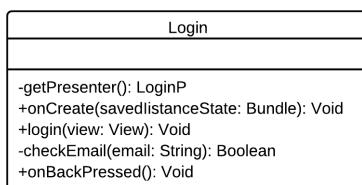


Figura 37: Classe com.woty.android.view.Login

#### 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.

### 5.3.3 com.woty.android.view.Home

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

Figura 38: 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.WorkgroupsRanks`
- `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.

### 5.3.4 com.woty.android.view.ModProfile

ModProfile
-textPassword: EditText
-getPresenter(): ModProfileP
+onCreate(savedInstanceState: Bundle): Void
+submit(requestCode: Int, resultCode: Int, intent: Intent): Void
#onResume(): Void

Figura 39: 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.

### 5.3.5 com.woty.android.view.PendingQuests

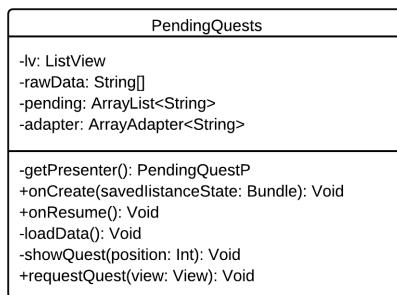


Figura 40: Classe com.woty.android.view.PendingQuests

## 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`
- `com.woty.android.presenter.C2dmMessageReceiver`

## 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 loadData()`  
Metodo che carica i dati da visualizzare dal server ed effettua la concatenazione delle stringhe presenti nell'array `rawData` settando così l'array `pending`.
- + `void requestQuest(View view)`  
Metodo invocato alla pressione del pulsante "Richiedi Quest" che causa la richiesta di nuove quest da svolgere al server.

### 5.3.6 com.woty.android.view.SolveQuest

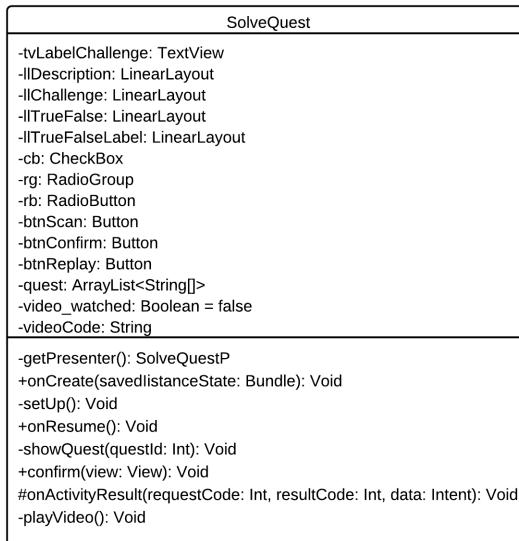


Figura 41: Classe com.woty.android.view.SolveQuest

#### 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.
- **Button btnScan**  
Button che permette la scansione del QR-Code nel caso la quest lo richieda.
- **Button btnReplay**  
Button che permette la ripetizione del video nel caso la sfida sia una sfida con video.
- **Button btnConfirm**  
Button che permette la conferma della risposta alla quest.
- **boolean video\_watched = false**  
*Flag* che verifica se il video è stato guardato dall'utente.
- **String VideoCode**  
Contiene il codice per il video di *Youtube*.
- **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 playVideo()**  
Metodo che lancia la visualizzazione del video, nel caso di una quest con video, utilizzando l'applicazione *Youtube* del dispositivo.
- + **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.

### 5.3.7 com.woty.android.view.WorkgroupsRanks

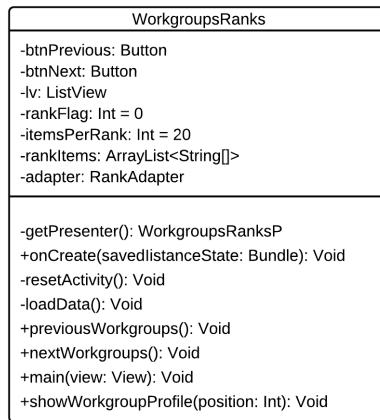


Figura 42: Classe com.woty.android.view.WorkgroupsRanks

#### Funzione della classe

Classe che si occupa della visualizzazione della finestra che espone la classifica dei workgroup del sistema. Per limitare la richiesta di dati vengono visualizzati solo 20 workgroup 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 la classifica degli utenti presenti all'interno dei vari workgroup.

#### Relazioni con altre classi

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

#### 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 itemsPerRank = 20`  
Intero che indica il numero massimo di workgroup visualizzabili nella listview.
- `ArrayList<String[]> rankItems = new ArrayList<String[]>()`  
`ArrayList` che contiene i dati riguardanti i workgroup presenti nella classifica che arrivano dal server. Ogni elemento corrisponde ad un workgroup e contiene un array con i dati dello stesso.
- `RankAdapter adapter`  
adapter della lista dei workgroup in classifica.

## 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 WorkgroupsRanksP 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.
- + `boolean loadData()`  
Metodo che carica dal server i dati da visualizzare nella classifica e setta quindi l'attributo `rankItems`.
- + `void nextWorkgroups()`  
Metodo invocato alla pressione del pulsante `btnNext` che provoca la richiesta al server dei 20 workgroup successivi nella classifica, sovrascrivendo l'array `rankItems`, e l'aggiornamento della lista con i nuovi valori.
- # `void previousWorkgroups()`  
Metodo invocato alla pressione del pulsante `btnPrevious` che provoca la richiesta al server dei 20 workgroup precedenti nella classifica, sovrascrivendo l'array `rankItems`, e l'aggiornamento della lista con i nuovi valori.
- + `myPosition(View view)`  
Metodo invocato alla pressione del pulsante "Mio dipartimento" utilizzato per visualizzare nella lista la zona di classifica contenente il workgroup dell'utente corrente.
- + `void main()`  
Metodo richiamato alla pressione del bottone "Principale" e torna a visualizzare i primi 20 workgroup in classifica.
- + `void showWorkgroupProfile(int position)`  
Metodo richiamato alla selezione di un workgroup nella classifica che provoca l'apertura di una finestra `UsersRanks` con la classifica specifica del workgroup in posizione `position`.

### 5.3.8 com.woty.android.view.UsersRanks

UsersRanks	
-btnPrevious: Button	
-btnNext: Button	
-lv: ListView	
-rankFlag: Int = 0	
-itemsPerRank: Int = 20	
-workgroupId: Int = -1	
-rankItems: ArrayList<String[]>	
-adapter: RankAdapter	
-getPresenter(): UsersRanksP	
+onCreate(savedInstanceState: Bundle): Void	
-resetActivity(): Void	
-loadData(): Void	
+previousPlayers(): Void	
+nextPlayers(): Void	
+myPosition(view: View): Void	
+main(view: View): Void	
+showPlayerProfile(position: Int): Void	

Figura 43: Classe com.woty.android.view.UsersRanks

#### Funzione della classe

Classe che si occupa della visualizzazione della finestra che espone la classifica degli utenti all'interno di un workgroup. 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.UsersRanksP
  - com.woty.android.exceptions.InvalidArgumentException
  - com.woty.android.exceptions.NotLoggedException
- È utilizzata dalle classi:
  - com.woty.android.view.WorkgroupsRanks
  - com.woty.android.presenter.UsersRanksP

#### 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 workgroupId = 1`  
Identificativo del workgroup collegato alla classifica.
- `int itemsPerRank = 20`  
Intero che indica il numero massimo di giocatori visualizzabili nella listview.
- `ArrayList<String[]> rankItems = new ArrayList<String[]>()`  
`ArrayList` che contiene i dati riguardanti i giocatori presenti nella classifica che arrivano dal server. Ogni elemento corrisponde ad un utente e contiene un array con i dati dello stesso.
- `RankAdapter adapter`  
`adapter` della lista dei giocatori in classifica.

## 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 UsersRanksP 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.
- + `boolean loadData()`  
Metodo che carica dal server i dati da visualizzare nella classifica e setta quindi l'attributo `rankItems`.
- + `void myPosition(View view)`  
Metodo utilizzato per visualizzare nella lista la zona di classifica contenente l'utente corrente.
- + `void main()`  
Metodo richiamato alla pressione del bottone "Principale" e torna a visualizzare i primi 20 utenti in classifica.
- + `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.

### 5.3.9 com.woty.android.view.RankAdapter

RankAdapter
- <b>items</b> : ArrayList<String[]> - <b>context</b> : Context
+ <b>RankAdapter(context: Context, textViewResourceId: Int, itemsArray: ArrayList&lt;String[]&gt;)</b> + <b>getView(position: Int, convertView: View, parent: ViewGroup): View</b>

Figura 44: Classe com.woty.android.view.RankAdapter

#### Funzione della classe

Classe che ridefinisce un generico  `ArrayAdapter<String[]>` per adattare le  `ListView` presenti nelle classifiche del sistema.

#### Relazioni con altre classi

- Estende la classe  `android.widget.ArrayAdapter`.
- È utilizzata dalle classi:
  - `com.woty.android.view.WorkgroupsRanks`
  - `com.woty.android.view.UsersRanks`

#### Attributi

- **ArrayList<String[]> items**  
     La lista di *item* da adattare nella  `ListView`.
- **Context context**  
     Contesto dell'applicazione.

#### Metodi

- + **RankAdapter(Context context, int textViewResourceId, ArrayList<String[]> itemsArray)**  
     Costruttore pubblico della classe.
- + **View getView(int position, View convertView, ViewGroup parent)**  
     Metodo ridefinito dalla classe  `ArrayAdapter` richiamato ogni volta in cui si procede con la visualizzazione di un *item* della lista. Permette di modificare la visualizzazione di determinati oggetti.

### 5.3.10 com.woty.android.view.PlayerProfile

PlayerProfile
- <b>user</b> : String[]
- <b>getPresenter()</b> : PlayerProfileP + <b>onCreate(savedInstanceState: Bundle)</b> : Void # <b>onResume()</b> : Void + <b>showProfile(u: User)</b> : Void

Figura 45: 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.UsersRanks`
  - `com.woty.android.presenter.PlayerProfileP`

### Attributi

- `String[] user`  
Lista dei dati da visualizzare sull'utente selezionato.

### Metodi

- + `void onCreate(Bundle savedInstanceState)`  
Metodo che viene richiamato alla creazione della finestra. Al suo interno viene settato il layout `R.layout.player_profile`.
- `final PlayerProfileP getPresenter()`  
Metodo che ritorna ed effettua il *cast* dell'oggetto `presenter`.
- `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.

#### 5.3.11 com.woty.android.view.Help

Help
<code>-helpList: ListView</code> <code>-backButton: ImageButton</code> <code>-closeButton: Button</code> <code>-context: Context</code>
<code>+Help(context: Context)</code> <code>+onCreate(savedInstanceState: Bundle): Void</code> <code>-initRoot(): Void</code> <code>-getBackButton(): Void</code> <code>-getCloseButton(): Void</code> <code>+onBackPressed(): Void</code>

Figura 46: 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.

### 5.3.12 com.woty.android.view.About

About
-closeButton: Button
+About(context: Context)
+onCreate(savedInstanceState: Bundle): Void

Figura 47: 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.

### 5.3.13 com.woty.android.view.ShowNotification

#### Funzione della classe

Classe che si occupa della visualizzazione, in una finestra dedicata, delle notifiche arrivate all'utente.

#### Relazioni con altre classi

- Estende la classe `android.app.Activity`.
- È utilizzata dalle classi:
  - `com.woty.android.presenter.C2dmRegistrationReceiver`

## Metodi

+ void **onCreate(Bundle savedInstanceState)**

Metodo che viene richiamato alla creazione della finestra. Imposta il layout della finestra a R.layout.show\_notification.

## 5.4 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.

### 5.4.1 com.woty.android.presenter.Presenter



Figura 48: 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.Configuration
  - com.woty.android.model.User
  - 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.WorkgroupsRanksP
  - com.woty.android.presenter.UsersRanksP
  - 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.

- static final boolean loadConfig(Context context)
    Metodo che carica le configurazioni del file config.xml e inizializza i campi della classe
    com.woty.android.model.Configuration.
```

### 5.4.2 com.woty.android.presenter.LoginP

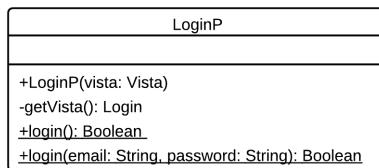


Figura 49: Classe com.woty.android.presenter.LoginP

## 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.
```

#### 5.4.3 com.woty.android.presenter.HomeP

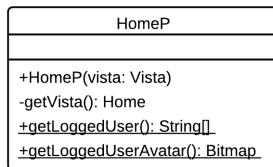


Figura 50: Classe com.woty.android.presenter.HomeP

##### 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.model.User
  - com.woty.android.view.Vista
  - com.woty.android.view.Home
  - 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 getVista()**  
 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.

#### 5.4.4 com.woty.android.presenter.PlayerProfileP

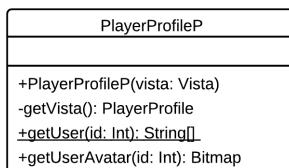


Figura 51: Classe com.woty.android.presenter.PlayerProfileP

##### 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.model.User`
  - `com.woty.android.view.Vista`
  - `com.woty.android.view.PlayerProfile`
  - `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.

### 5.4.5 com.woty.android.presenter.ModProfileP

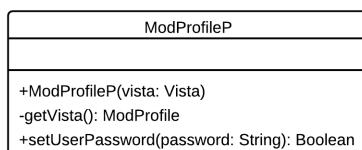


Figura 52: Classe com.woty.android.presenter.ModProfileP

### 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.model.User`
  - `com.woty.android.view.Vista`
  - `com.woty.android.view.ModProfile`
  - `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 setPassword(String password) throws NotLoggedException`  
Metodo che modifica la password dell'utente corrente.

### 5.4.6 com.woty.android.presenter.PendingQuestsP

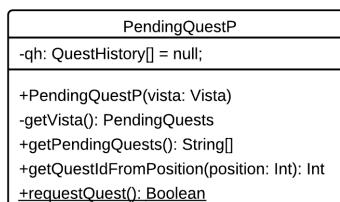


Figura 53: Classe com.woty.android.presenter.PendingQuestsP

## 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.model.Quest`
  - `com.woty.android.model.QuestHistory`
  - `com.woty.android.view.Vista`
  - `com.woty.android.view.PendingQuests`
  - `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

- + `PendingQuestP(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.

+ static final boolean requestQuest() throws NotLoggedException
    Metodo che richiede l'assegnamento di una quest dal server.
```

#### 5.4.7 com.woty.android.presenter.SolveQuestP

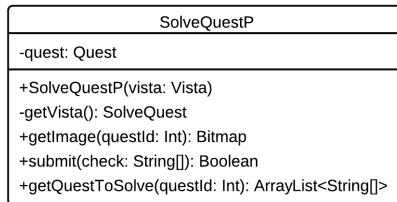


Figura 54: Classe com.woty.android.presenter.SolveQuestP

#### 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.Resource
  - com.woty.android.model.User
  - com.woty.android.model.Challenge
  - com.woty.android.model.TrueFalseChallenge
  - com.woty.android.model.ComboChallenge
  - com.woty.android.model.MultiOptionChallenge
  - com.woty.android.model.QRChallenge
  - com.woty.android.model.Description
  - com.woty.android.model.TextDescription
  - com.woty.android.model.ImageDescription
  - com.woty.android.model.VideoDescription
  - com.woty.android.model.Quest
  - com.woty.android.model.Session
  - 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

### + `SolveQuestP(Vista vista) throws InvalidArgumentException`

Costruttore della classe che riceve la vista associata.

### - `final SolveQuest getView()`

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[]>`.

## 5.4.8 com.woty.android.presenter.WorkgroupsRanksP



Figura 55: Classe com.woty.android.presenter.WorkgroupsRanksP

## Funzione della classe

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

## Relazioni con altre classi

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

## Metodi

- + `WorkgroupsRanksP(Vista vista) throws InvalidArgumentException`  
Costruttore della classe che riceve la vista collegata.
- `final WorkgroupsRanks getVista()`  
Metodo che ritorna ed effettua il *cast* della classe `WorkgroupsRanks` collegata al presenter.
- + `static final ArrayList<String[]> getRank(int start) throws NotLoggedException`  
Metodo che ottiene dal server un Rank, sotto forma di array di stringhe contenenti le informazioni sui workgroup presenti in classifica passando la posizione iniziale richiesta.

### 5.4.9 com.woty.android.presenter.UsersRanksP

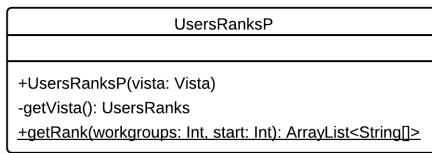


Figura 56: Classe com.woty.android.presenter.UsersRanksP

## Funzione della classe

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

## Relazioni con altre classi

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

## Metodi

- + `UsersRanksP(Vista vista) throws InvalidArgumentException`  
Costruttore della classe che riceve la vista collegata.
- `final UsersRanks getVista()`  
Metodo che ritorna ed effettua il *cast* della classe `UsersRanks` 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 la posizione iniziale richiesta e il workgroup di appartenenza.

#### 5.4.10 com.woty.android.presenter.C2dmMessageReceiver

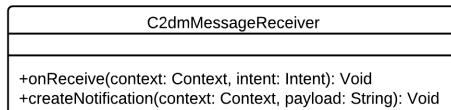


Figura 57: Classe com.woty.android.presenter.C2dmMessageReceiver

##### 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`.
- Utilizza le classi:
  - `com.woty.android.view.PendingQuests`

##### 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.

#### 5.4.11 com.woty.android.presenter.C2dmRegistrationReceiver

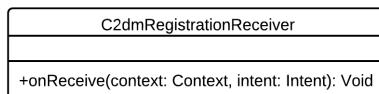


Figura 58: Classe com.woty.android.presenter.C2dmRegistrationReceiver

##### 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.view.ShowNotification`
  - `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.

## 5.5 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.

### 5.5.1 com.woty.android.util.JParser

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

Figura 59: Classe com.woty.android.remote.JParser

#### Funzione della classe

Classe astratta che fornisce le funzionalità di parser<sup>g</sup> 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.Workgroup
  - com.woty.android.model.Challenge
  - com.woty.android.model.Description
  - com.woty.android.model.Quest
  - com.woty.android.model.QuestHistory
  - com.woty.android.model.WorkgroupsRank
  - com.woty.android.model.UsersRank

#### 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.

## 5.6 Specifica package exceptions

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

### 5.6.1 com.woty.android.exceptions.NotLoggedException

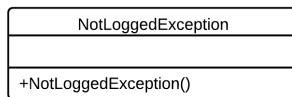


Figura 60: Classe com.woty.android.exceptions.NotLoggedException

#### Funzione della classe

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

#### Metodi

- + `NotLoggedException()`

Costruttore pubblico della classe.

### 5.6.2 com.woty.android.exceptions.InvalidArgumentException

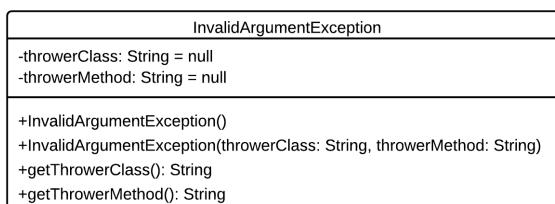


Figura 61: Classe com.woty.android.exceptions.InvalidArgumentException

#### 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`.

## 6 Tracciamento Requisiti-Componenti-Classi

Per quanto riguarda il tracciamento si rimanda a "*Tracciamento relazioni componenti - requisiti*", sezione 6 della Specifica Tecnica (documento allegato *SpecificaTecnica\_v3.pdf*).