PROGETTAZIONE

NICOLA MORETTO (MATR. 578258)

1 novembre 2012

Il documento riporta le informazioni di progettazione riguardanti l'interfaccia grafica per la visualizzazione e la navigazione dei contenuti.

Versione	Data	Modifiche
0.1	15-10-2012	Stesura iniziale del documento.
0.2	17-10-2012	Redatto il capitolo 2.
0.3	18-10-2012	Redatte le sezioni 3.3, 4.3 e 4.4.
0.4	19-10-2012	Redatti i capitoli 3 e 4.
0.5	20-10-2012	Revisione del documento.
1.0	20-10-2012	Pubblicazione della prima versione.
1.1	22-10-2012	Aggiunte la sezioni 3.4 e 3.6.
1.2	23-10-2012	Aggiornate le sezioni 3.2 e 3.3.
1.3	24-10-2012	Aggiornate le sezioni 3.1 e 3.5.
1.4	26-10-2012	Aggiornato il capitolo 4.
1.5	27-10-2012	Aggiunti i capitoli 6 e 7.
2.0	28-10-2012	Pubblicazione della seconda versione.
2.1	29-10-2012	...
2.2	30-10-2012	...

Tabella 1: Registro delle modifiche

INDICE

ELENCO DELLE FIGURE

1 INTRODUZIONE

1.1 CONVENZIONI

La nomenclatura adottata per i package e le classi è il *CamelCase*. L'identificatore di ciascuna sottoclasse adotta come prefisso le lettere maiuscole presenti nel nome della rispettiva classe base (nel medesimo ordine di apparizione).

1.2 RIFERIMENTI INFORMATIVI

- Analisi dei requisiti (*analisi_dei_requisiti_1.0* allegata alla presente documentazione);
- Sistema di classificazione (*sistema_di_classificazione_2.0* allegato alla presente documentazione).

2 ARCHITETTURA

2.1 ARCHITETTURA GENERALE

L'architettura del sistema software rispecchia il design pattern architetturale MVC, che prevede e garantisce la separazione delle tre componenti fondamentali del sistema:

MODEL Racchiude i dati e le informazioni dell'applicazione e definisce le modalità di accesso e fruizione degli stessi da parte delle altre componenti (*Controller* e *View*).

VIEW Rappresenta l'interfaccia grafica mediante la quale vengono visualizzate le informazioni e i dati conservati nel *Model* e l'utente può interagire con il sistema. La rilevazione dell'avvenuta interazione dell'utente è responsabilità di tale componente, mentre la gestione della reazione è demandata al *Controller*.

CONTROLLER Incorpora la logica di controllo dell'applicazione, inizializzando il sistema e traducendo l'interazione dell'utente con l'interfaccia grafica (*View*) in operazioni sui dati (*Model*).

2.2 COMPONENTI ARCHITETTURALI

2.2.1 *Componente Model*

La componente *model* conserva tutte i tipi di informazioni connessi alla ricerca:

- gli ambiti (etichette, frasi);
- i criteri (termini di ricerca, entità);
- i filtri (argomento, data di pubblicazione, emozioni, ...);
- i risultati (contenuti informativi).

2.2.2 *Componente View*

La componente *view* rappresenta l'interfaccia grafica mediante la quale l'utente interagisce con il sistema per effettuare una ricerca, raffinarne i criteri o consultarne i risultati.

LIVELLI Essa è organizzata in quattro livelli (o strati) distinti, che includono rispettivamente:

- gli strumenti e le informazioni connesse alla ricerca (barra di ricerca, etichette, entità, ...);
- i filtri di ricerca;
- i risultati della ricerca (proprietà e relazioni dei contenuti informativi;
- la cronologia (organizzazione temporale dei contenuti).

2.2.3 *Componente Controller*

La componente *controller* gestisce l'interazione dell'utente con l'interfaccia grafica e le operazioni connesse alla ricerca e alla visualizzazione dei contenuti informativi, tra cui:

- la configurazione dei criteri di ricerca (selezione di un'accezione di un'etichetta, gestione delle entità);
- il reperimento dei contenuti informativi corrispondenti ai criteri di ricerca;
- la gestione dei filtri di ricerca;
- l'aggiornamento dei risultati di ricerca visualizzati a fronte di modifiche alle entità e ai filtri di ricerca;
- ...

3 COMPONENTE MODEL

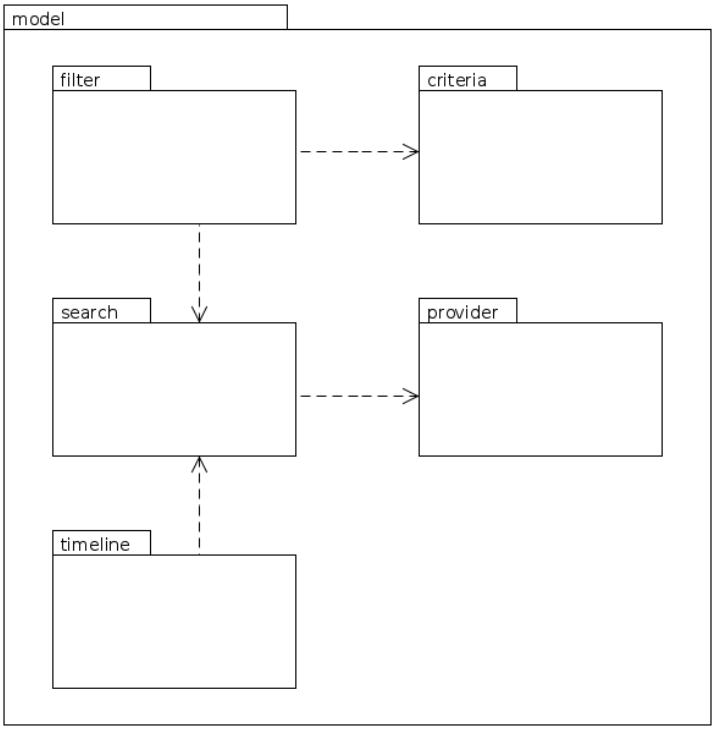

Figura 1: Diagramma del package *model*

Questo capitolo illustra i componenti del *Model*, per ciascuno dei quali è indicato il nome della classe accompagnata da un'identificazione sintetica, separati dal carattere '|' (separatore verticale).

3.1 PACKAGE MODEL

3.1.1 *Content* | *Contenuto informativo*

La classe *Content* rappresenta un generico contenuto informativo pubblicato dagli utenti nella piattaforma.

3.1.2 *Metadata* | *Metadati dei contenuti*

La componente *Metadata* rappresenta una meta-informazione generica associata ad un contenuto informativo.

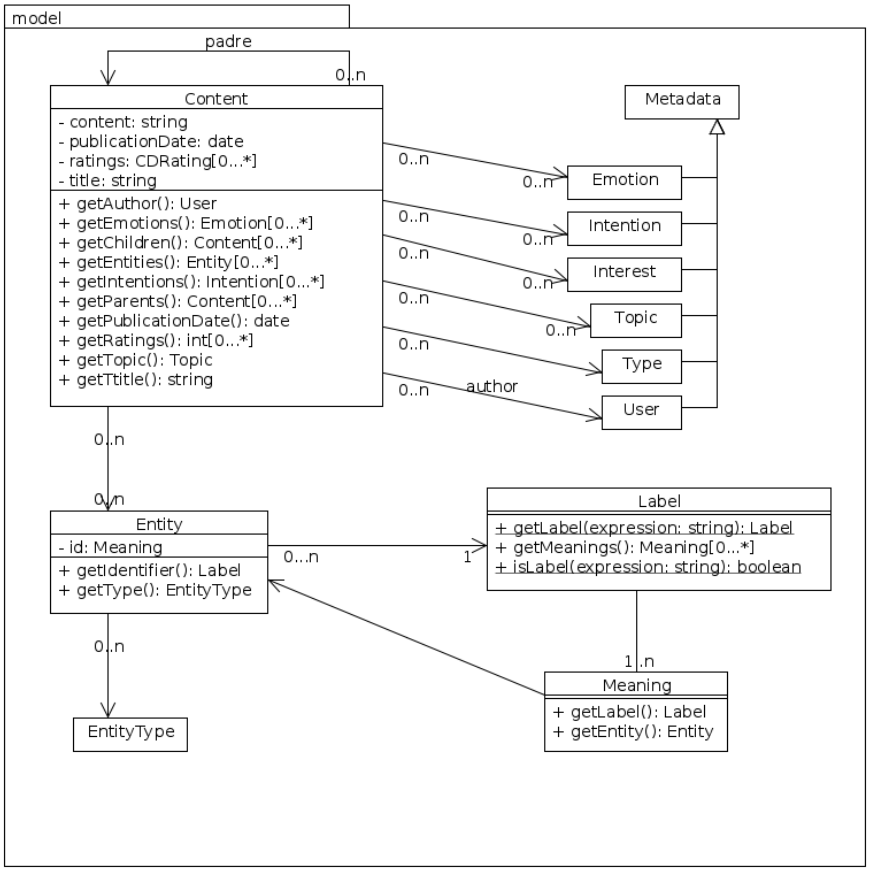

Figura 2: Diagramma delle classi del package *model*

3.1.3 *Emotion* | *Emozioni*

La classe *Emotion* rappresenta un'emozione associabile ad un contenuto ed estende la classe *Metadata*.

3.1.4 *Intention* | *Intenzione*

La classe *Intention* rappresenta un'intenzione associabile ad un contenuto ed estende la classe *Metadata*.

3.1.5 *Interest* | *Interesse*

La classe *Interest* rappresenta gli interessi cui è associabile un contenuto ed estende la classe *Metadata*.

3.1.6 *Topic* | *Argomento*

La classe *Topic* rappresenta un argomento cui può appartenere un contenuto ed estende la classe *Metadata*.

3.1.7 *Type* | *Tipo di contenuto*

La classe *Intention* rappresenta un'intenzione associabile ad un contenuto ed estende la classe *Metadata*.

3.1.8 *User* | *Autore*

La classe *User* rappresenta l'utente che ha pubblicato il contenuto ed estende la classe *Metadata*.

3.1.9 *Entity* | *Entità del dominio*

La classe modella un'entità del dominio della piattaforma, cui è associata un'etichetta - in una specifica accezione - che la identifica univocamente nell'ambito della piattaforma (*Label*).

3.1.10 *EntityType* | *Tipo di entità*

La classe rappresenta i tipi di entità del dominio (*Entity*).

3.1.11 *Label* | *Etichetta del dizionario*

La classe modella le etichette del dizionario della piattaforma, ciascuna delle quali può possedere molteplici accezioni (*Meaning*).

3.1.12 *Meaning* | *Accezione di un'etichetta*

La classe modella un'accezione di un'etichetta (*Label*), che riferisce un'entità del dominio (*Entity*).

3.2 PACKAGE MODEL.CRITERIA

Ciascun contenuto può essere classificato in accordo a differenti criteri, ciascuno dei quali ne prende in esame una proprietà (autore, data di pubblicazione, tipo), un criterio di classificazione (emozioni, intenzioni, ...) o un informazione contestuale alla ricerca (attinenza, ...) differenti.

3.2.1 *CriteriaModel* | *Gestione dei criteri di classificazione*

La classe *CriteriaModel* rappresenta l'interfaccia del package *model.criteria*.

3.2.2 *Criterion* | *Criteri di classificazione*

Tale componente rappresenta un'interfaccia per le classi dei criteri di classificazione.

3.2.3 *CList* | *Criterio basato su una lista di valori*

La classe *CList* rappresenta un criterio di classificazione basato su una lista di valori, ciascuno dei quali può essere ammissibile o bloccato.

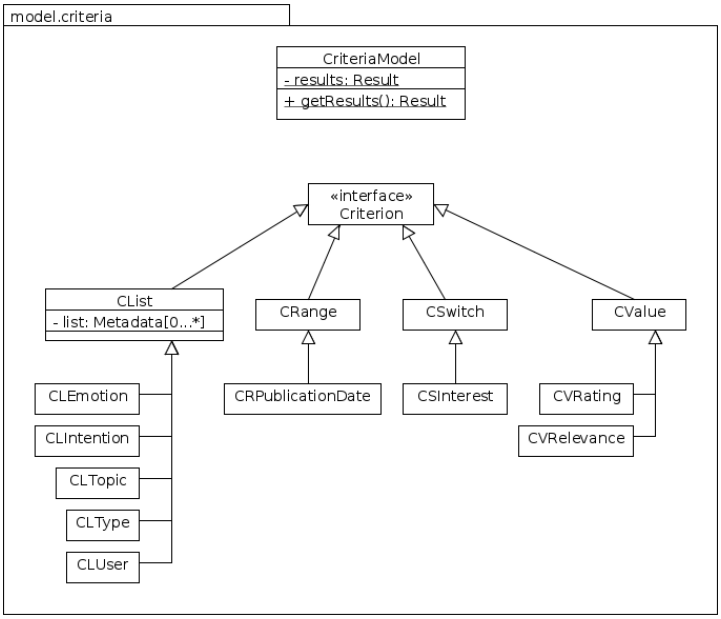

Figura 3: Diagramma delle classi del package *model.criteria*

Ciascuna istanza viene utilizzata in combinazione con un filtro *FList*, fornendo i valori e i parametri di configurazione iniziale.

3.2.4 *CRange* | *Criterio basati su un intervallo di valori*

La classe *CRange* rappresenta un criterio di classificazione basato su un intervallo di valori.

Ciascuna istanza viene utilizzata in combinazione con un filtro *FRange*, fornendo i valori e i parametri di configurazione iniziale.

3.2.5 *CSwitch* | *Criterio a doppio stato*

La classe *CSwitch* rappresenta un criterio di classificazione che l'utente può semplicemente decidere di attivare o disattivare.

Ciascuna istanza viene utilizzata in combinazione con un filtro *FSwitch*, fornendo i valori e i parametri di configurazione iniziale.

3.2.6 *CValue* | *Filtro basato su una soglia di valore*

La classe *CValue* rappresenta un criterio di classificazione basato su una soglia di valore.

Ciascuna istanza viene utilizzata in combinazione con un filtro *FValue*, fornendo i valori e i parametri di configurazione iniziale.

3.2.7 *CLEmotion* | *Emozione*

La classe *CLEmotion* rappresenta un criterio di classificazione a lista di valori basato sulle emozioni.

3.2.8 *CLIntention* | *Intenzione*

La classe *CLIntention* rappresenta un criterio di classificazione a lista di valori basato sulle intenzioni..

3.2.9 *CLTopic* | *Argomento*

La classe *CLTopic* rappresenta un criterio di classificazione a lista di valori basato sugli argomenti.

3.2.10 *CLType* | *Tipo di contenuto*

La classe *CLType* rappresenta un criterio di classificazione a lista di valori basato sui tipi di contenuto.

3.2.11 *CLUser* | *Utente*

La classe *CLUser* rappresenta un criterio di classificazione a lista di valori basato sugli autori dei contenuti.

3.2.12 *CRDate* | *Data di pubblicazione*

La classe *CRDate* rappresenta un criterio di classificazione ad intervallo di valori basato sulla data di pubblicazione.

3.2.13 *CSInterest* | *Interessi dell'utente*

La classe *CSInterest* rappresenta un criterio di classificazione ad interruttore basato sugli interessi dell'utente.

3.2.14 *CVRating* | *Giudizio*

La classe *CVRating* rappresenta un criterio di classificazione a soglia di valore basato sui giudizi espressi dagli utenti sui contenuti.

3.2.15 *CVRelevance* | *Attinenza*

La classe *CVRelevance* rappresenta un criterio di classificazione a soglia di valore basato sulla rilevanza dei contenuti rispetto ai criteri di ricerca immessi.

3.3 PACKAGE MODEL.FILTER

Ciascun filtro è associato ad un possibile criterio di classificazione (*Criterion*), che partiziona automaticamente l'insieme dei possibili valori in due sottoinsiemi: ammessi o bloccati.

Nella configurazione iniziale, tutti i valori possibili di una proprietà sono ammessi, mentre l'utente può intervenire secondo modalità differenti per alterare tale partizionamento (vedi sezione 4.3).

Il valore che ciascun contenuto (risultato di una ricerca) assume in relazione ad una proprietà associata ad un filtro può quindi appartenere ad uno dei

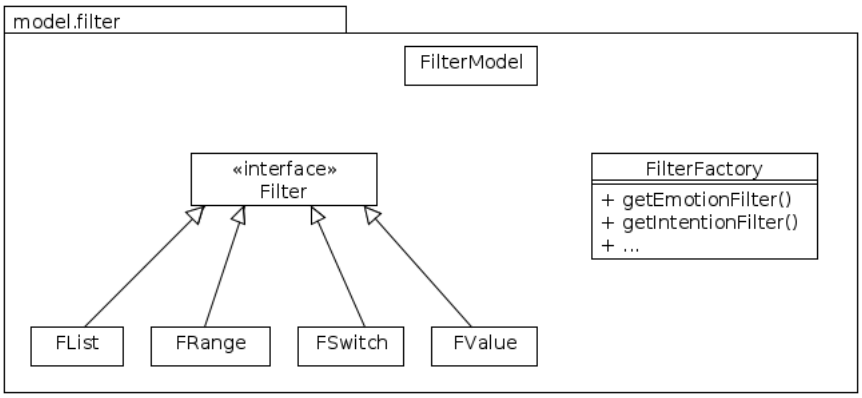

Figura 4: Diagramma delle classi del package *model.filter*

due sottoinsiemi: il contenuto viene mostrato solo se tutti i valori delle proprietà in questione risultano ammessi.

3.3.1 *FilterFactory* | *Creazione dei filtri*

La classe *FilterFactory* offre un'interfaccia per creare filtri (*Filter*) basati sui possibili criteri (*Criterion*) in modo trasparente rispetto al client.

	Criterion	Filter	FilterView
Argomento	CList	FList	FList
Attinenza	CValue	FValue	FValue
Autore	CList	FList	FList
Data di pub.	CRange	FRange	FRange
Emozioni	CList	FList	FList
Giudizi	CValue	FValue	FValue
Intenzioni	CList	FList	FList
Interessi	CSwitch	FSwitch	FSwitch
Tipo	CList	FList	FList

Tabella 2: Classi istanziate da *FilterFactory*

3.3.2 *FilterModel* | *Gestione dei filtri*

Tale componente rappresenta l'interfaccia del package *model.filter*, utile a esporre le funzionalità per l'istanziazione e la gestione dei filtri.

3.3.3 *Filter* | *Filtro di ricerca*

Tale componente rappresenta l'interfaccia dei filtri per il raffinamento dei risultati di una ricerca e viene implementata dalle classi che modellano le tipologie standard di filtri di ricerca (*FList*, *FRange*, *FSwitch* e *FValue*).

Ciascuna istanza di una sottoclasse concreta è associata ai risultati di una ricerca specifica (*Result*), su cui si applica il filtro corrispondente.

3.3.4 *FList* | *Filtro a lista di valori*

La classe *FList* modella un filtro basato su una lista di possibili valori, ciascuno dei quali può essere autorizzato o bloccato dall'utente, e ciascuna delle relative istanze è associata ad un criterio *CList*.

3.3.5 *FRange* | *Filtro ad intervallo di valori*

La classe *FRange* modella un filtro basato su un intervallo di valori ordinati (numeri, date, ...) e ciascuna istanza è associata ad un criterio *CRange*.

Siano inf e sup rispettivamente l'estremo inferiore e superiore dell'intervallo: risultano dunque ammessi tutti e soli i valori v tali che $v \in [inf, sup]$.

All'utente è consentito scegliere i valori desiderati di inf e sup tali che:

- sup e inf siano valori validi;
- $inf \leq sup$;
- se è definito un valore attuale *current*, allora deve valere $min \leq current \leq max$.

Se l'insieme dei valori ordinati prevede un minimo min e/o un massimo max si aggiungono le seguenti condizioni:

- $sup \leq max$;
- $inf \geq min$.

3.3.6 *FSwitch* | *Filtro a doppio stato*

La classe *FSwitch* modella un filtro il cui partizionamento è predefinito e invariabile e che l'utente può solamente abilitare o disattivare. Ciascuna istanza è associata ad un criterio *CSwitch*.

3.3.7 *FValue* | *Filtro a soglia di valore*

La classe *FValue* rappresenta un filtro basato su una soglia di valore, ciascuna delle cui istanze è associata ad un criterio *CValue*.

Sia $value$ il valore scelto: in tal caso risultano ammessi tutti e soli i valori x validi tali che $x \geq value$. Se l'insieme prevede un minimo min e/o un massimo max dev'essere soddisfatta anche la condizione $min \leq x \leq max$.

3.4 PACKAGE MODEL.PROVIDER

Le classi del package *model.provider* forniscono un livello di astrazione per accedere alla base di dati e interrogarla al fine di reperire i contenuti informativi corrispondenti ai criteri di ricerca immessi e all'ambito specificato.

Dal momento che tali operazioni verranno affidate a componenti terze, tuttora oggetto di analisi e valutazione, le scelte progettuali illustrate di seguito cercano di tenere conto di tale incertezza.

3.4.1 *Provider* | *Motore di ricerca*

La componente rappresenta l'interfaccia standard implementata dai fornitori di ricerca (*PLabel*, *PFullText*).

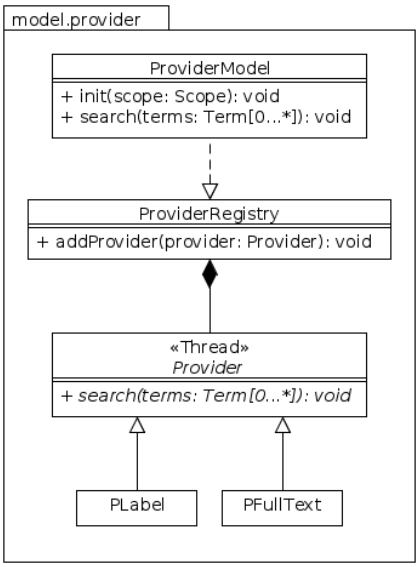

Figura 5: Diagramma delle classi del package *model.provider*

3.4.2 *ProviderModel* | *Gestore motori di ricerca*

La classe *ProviderModel* rappresenta l'interfaccia del package *model.provider* e fornisce i metodi per accedere alle funzionalità di ricerca.

3.4.3 *ProviderRegistry* | *Gestione dei fornitori di ricerca*

La classe *ProviderRegistry* gestisce gli oggetti di tipo *Provider*.

3.4.4 *PLabel* | *Ricerca delle etichette*

La classe *PLabel* fornisce le funzionalità necessaria a cercare i termini di ricerca tra le etichette assegnate ai contenuti, considerandone le specifiche accezioni.

Essa implementa l'interfaccia *Provider* e le sue istanze vengono gestite da *ProviderRegistry*.

3.4.5 *PFullText* | *Ricerca dei contenuti informativi*

La classe *PFullText* permette di cercare la presenza dei termini di ricerca nel titolo, nel corpo o in altri campi di un contenuto informativo.

Essa implementa l'interfaccia *Provider* e le sue istanze vengono gestite da *ProviderRegistry*.

3.5 PACKAGE MODEL.SEARCH

Il package *model.search* raccoglie e gestisce le informazioni inerenti i parametri e i risultati di una ricerca.

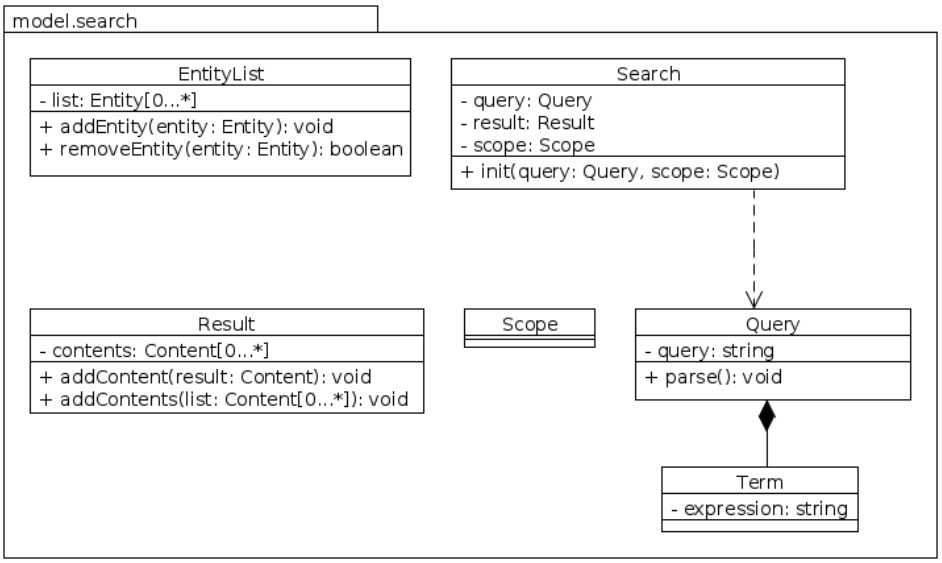

Figura 6: Diagramma delle classi del package *model.search*

3.5.1 *Search* | *Gestione della ricerca*

Tale componente rappresenta l'interfaccia del package *model.search*.

3.5.2 *EntityList* | *Lista delle entità cercate*

La classe *EntityList* rappresenta la lista delle entità individuate a partire dai termini di ricerca corrispondenti ad etichette del dizionario.

3.5.3 *Query* | *Query di ricerca*

La classe *Query* rappresenta la query di ricerca, ossia la stringa inserita dall'utente nella barra di ricerca (*SearchBar*) e contenente una lista di termini o espressioni separati da virgola.

La classe fornisce i metodi per effettuare l'analisi sintattica della stringa al fine di estrapolare i termini (*Term*) da cercare e stabilire quali di essi siano etichette o frasi.

3.5.4 *Result* | *Risultati della ricerca*

La classe *Result* rappresenta l'insieme dei risultati di una ricerca (*Content*).

3.5.5 *Scope* | *Ambito di ricerca*

La classe *Scope* rappresenta un ambito di ricerca e ne rende disponibile la lista completa.

3.5.6 *Term* | *Termine di ricerca*

La classe *Term* rappresenta un generico termine di ricerca, estrapolato dalla query (*Query*).

3.6 PACKAGE MODEL.TIMELINE

Il package *model.timeline* raccoglie le informazioni inerenti l'organizzazione e la visualizzazione cronologica dei contenuti.

3.6.1 *Timeline* | *Linea del tempo*

La classe *Timeline* rappresenta la linea del tempo, suddivisa in intervalli temporali *TimeUnit* di ampiezza fissa - personalizzabile dall'utente - in cui vengono collocati i risultati della ricerca o i contenuti di una discussione.

3.6.2 *TimeUnit* | *Unità temporale*

La classe *TimeUnit* rappresenta un singolo intervallo di tempo e raccoglie i risultati di ricerca o i contenuti di una discussione pubblicati nel periodo corrispondente.

4 COMPONENTE VIEW

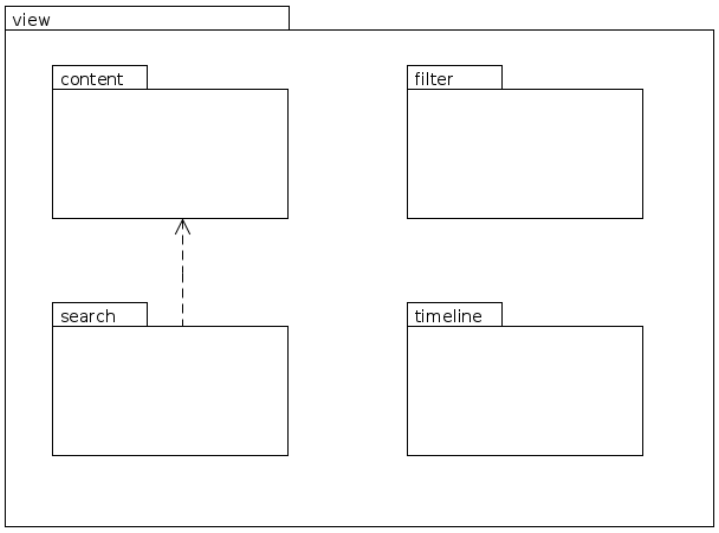

Figura 7: Diagramma del package *view*

Questo capitolo illustra i componenti del *View*, per ciascuno dei quali è indicato il nome della classe accompagnata da un'identificazione sintetica, separati dal carattere '|' (separatore verticale).

4.1 PACKAGE VIEW

4.1.1 *MainWindow* | *Finestra principale*

La classe *MainWindow* rappresenta la finestra principale all'interno del quale vengono inseriti e opportunamente collocati tutti gli elementi grafici dell'interfaccia, tra cui gli strumenti di ricerca, i filtri e i contenuti.

4.2 PACKAGE VIEW.CONTENT

4.2.1 *ContentView* | *Contenuto informativo*

Si tratta del componente grafico deputato a rappresentare graficamente un contenuto (*Content*) e le relative informazioni (in formato grafico o testuale).

Informazioni essenziali

Le informazioni essenziali sono utili per una immediato e preciso inquadramento del contenuto da parte dell'utente al fine di stabilirne la rilevanza soggettiva:

AUTORE L'autore del contenuto è l'utente (*CLUser*) che lo ha pubblicato all'interno della piattaforma e viene rappresentato testualmente mediante il suo *nome utente* o *proprio*.

Figura 8: Informazioni essenziali di un contenuto

ATTINENZA Il grado di attinenza di un contenuto rispetto ai criteri di ricerca (*CVRelevance*) corrisponde - in percentuale - al rapporto tra le entità assegnate e quelle cercate. Tale informazione viene rappresentata graficamente variando proporzionalmente la *dimensione* dell'elemento grafico.

DATA DI PUBBLICAZIONE La data di pubblicazione del contenuto viene indicata testualmente.

TIPO Il tipo di contenuto (*CLType*) viene rappresentato graficamente variando la forma[1] dell'elemento grafico.

TITOLO Il titolo assegnato al contenuto viene rappresentata in formato testuale.

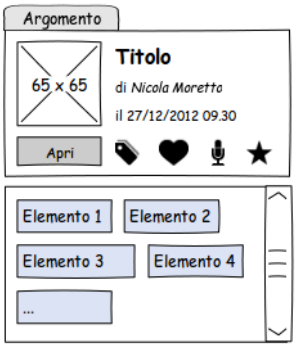

Figura 9: Informazioni aggiuntive di un contenuto

informazioni aggiuntive

Le informazioni aggiuntive forniscono all'utente dettagli utili per approfondire l'esame di un contenuto.

ARGOMENTO Ciascun contenuto può riferirsi al più ad un argomento (*CLTopic*), rappresentabile in maniera grafica (colore o simbolo) o testuale.

EMOZIONI A ciascun contenuto possono essere associate diverse emozioni (*CLEmotion*), che esprimono lo stato d'animo dell'autore al momento della sua redazione o pubblicazione. La lista delle emozioni associate ad un contenuto viene visualizzata in formato testuale.

1 Le forme utilizzate sono elementari per garantire l'immediata riconoscibilità da parte di qualsiasi tipo di utente, evitando l'impiego di forme potenzialmente ambigue o ignote a seconda del suo profilo sociale, culturale, geografico, ...

ENTITÀ A ciascun contenuto possono essere assegnate delle etichette (*Label*), che riferiscono altrettante entità del dominio da visualizzare in formato testuale (mediante il rispettivo nome).

GIUDIZIO Il giudizio ((*CVRating*)) associato ad un contenuto è un'informazione rappresentabile in formato testuale o grafico.

INTENZIONI A ciascun contenuto possono essere associate delle intenzioni (*CLIntention*), che possano fornire delle linee guida interpretative ai lettori. La lista delle intenzioni associate ad un contenuto viene visualizzata in formato testuale.

4.2.2 *CVAnswer* | *Risposta*

La classe rappresenta il componente grafico che visualizza un contenuto di tipo *risposta* ed estende la classe *ContentView*.

4.2.3 *CVEvent* | *Evento*

La classe rappresenta il componente grafico che visualizza un contenuto di tipo *evento* ed estende la classe *ContentView*.

4.2.4 *CVMessage* | *Comunicazione privata*

La classe rappresenta il componente grafico che visualizza un contenuto di tipo *comunicazione privata* ed estende la classe *ContentView*.

4.2.5 *CVQuestion* | *Domanda*

La classe rappresenta il componente grafico che visualizza un contenuto di tipo *domanda* ed estende la classe *ContentView*.

4.2.6 *CVReview* | *Recensione*

La classe rappresenta il componente grafico che visualizza un contenuto di tipo *recensione* ed estende la classe *ContentView*.

4.2.7 *CVTalk* | *Discorso*

La classe rappresenta il componente grafico che visualizza un contenuto di tipo *discorso* ed estende la classe *ContentView*.

4.2.8 *CVThought* | *Pensiero*

La classe rappresenta il componente grafico che visualizza un contenuto di tipo *pensiero* ed estende la classe *ContentView*.

4.2.9 *EntityView* | *Entità del dominio*

La classe *EntityView* rappresenta il componente grafico deputato a visualizzare le informazioni essenziali associate ad un'entità e a consentire all'utente di:

- visualizzare la lista delle entità padre;
- visualizzare la lista delle entità figlie;
- sostituire l'entità corrente con un padre o un figlio.
- eliminare l'entità corrente.

4.3 PACKAGE VIEW.FILTER

Il package *view.filter* definisce le componenti grafiche per la visualizzazione e l'interazione dell'utente con i filtri di ricerca.

4.3.1 *FilterContainer* | *Livello filtri*

Il *FilterContainer* raccoglie e mantiene separati in un livello distinto gli elementi grafici di tipo *FilterView*, che forniscono all'utente gli strumenti per impostare i filtri di ricerca.

4.3.2 *FilterView* | *Filtro di ricerca*

Tale componente rappresenta l'interfaccia delle componenti grafiche per le classi standard di filtri per il raffinamento della ricerca (*FVList*, *FVRange*, *FVSwitch* e *FVValue*).

4.3.3 *FVList* | *Filtro con lista di valori*

Il componente grafico - associato alla classe *FList* - visualizza le liste dei valori ammissibili e bloccati e consente all'utente di:

- spostare un valore da una lista all'altra;
- azzerare il filtro, spostando automaticamente tutti i valori nella lista degli ammissibili.

4.3.4 *FVRange* | *Filtro con intervallo di valori*

Il componente grafico - associato alla classe *FRange* - permette all'utente di specificare l'estremo inferiore e superiore dell'intervallo dei valori ammissibili secondo le regole definite nella suddetta classe.

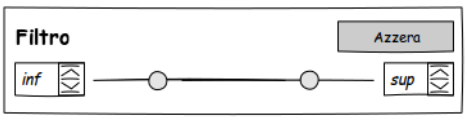

Figura 10: Filtro ad intervallo di valori

4.3.5 *FVSwitch* | *Filtro a doppio stato*

Il componente grafico - associato alla classe *FSwitch* - consente di abilitare o disattivare il filtro corrispondente o di visualizzarne lo stato corrente.

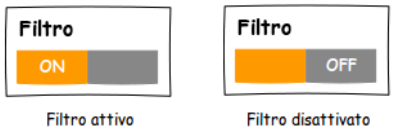

Figura 11: Filtro a doppio stato

4.3.6 *FVValue* | *Filtro con soglia di valore*

Il componente grafico - associato alla classe *FValue* - consente all'utente di modificare il valore della soglia associata al filtro corrispondente.

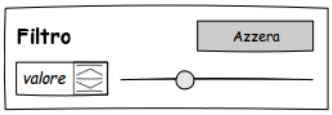

Figura 12: Filtro a soglia di valore

4.4 PACKAGE VIEW.SEARCH

4.4.1 *SearchContainer* | *Livello di ricerca*

Il *SearchContainer* raccoglie e mantiene separati in un livello distinto gli elementi grafici relativi alla ricerca.

4.4.2 *EntityListWidget* | *Elenco delle entità cercate*

La classe *EntityListWidget* rappresenta le componente grafica che visualizza le entità (*EntityView*) riferite dalle etichette e cercate tra i contenuti.

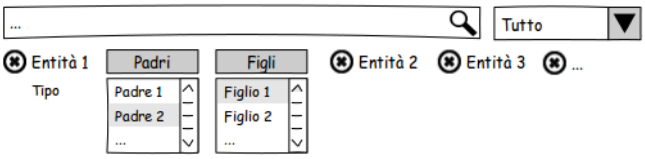

Figura 13: Visualizzazione dei dettagli di un'entità

4.4.3 *LabelEntityWidget* | *Accezioni di un'etichetta*

La classe *LabelEntityWidget* rappresenta il componente grafico che mostra - per ciascuna etichetta inclusa nei termini di ricerca e avente accezioni multiple - la lista delle entità (*EntityView*) riferite, consentendo all'utente di selezionare quella rispetto cui intenda procedere con la ricerca.

Figura 14: Selezione dell'accezione delle etichette

4.4.4 *SearchBar* | *Barra di ricerca*

La classe *SearchBar* rappresenta la barra di ricerca mediante la quale l'utente può inserire i termini di ricerca.

4.4.5 *SearchScopeSelector* | *Selettore dell'ambito di ricerca*

La classe *SearchScopeSelector* rappresenta il selettore dell'ambito di ricerca, che permette all'utente di circoscrivere la ricerca ad informazioni specifiche:

TUTTO
: Estende la ricerca a tutti i tipi di informazioni indicizzate o ricercabili all'interno della piattaforma.

ETICHETTE
: Limita la ricerca alle sole etichette assegnate ai contenuti.

FRASI
: Limita la ricerca alle informazioni presenti in un contenuto (titolo, corpo, ...).

4.5 PACKAGE VIEW.TIMELINE

4.5.1 *Timeline* | *Cronologia dei contenuti*

La classe *Timeline* rappresenta il componente grafico deputato alla visualizzazione cronologica dei contenuti e degli strumenti di navigazione.

4.5.2 *TimeSlot* | *Unità temporale*

La classe *TimeSlot* rappresenta il componente grafico deputato alla visualizzazione dei contenuti pubblicati in un certo intervallo di tempo (*TimeUnit*).

4.5.3 *TimeAxis* | *Asse temporale*

La classe *TimeAxis* rappresenta l'asse temporale, sul quale è indicata la scala utilizzata e sono forniti gli strumenti per la navigazione.

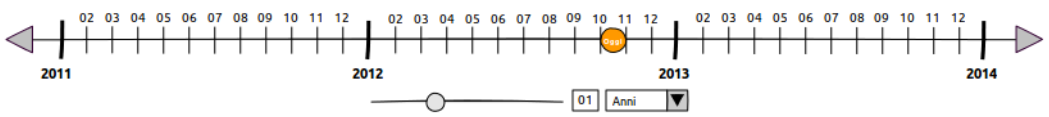

Figura 15: Asse temporale

5 COMPONENTE CONTROLLER

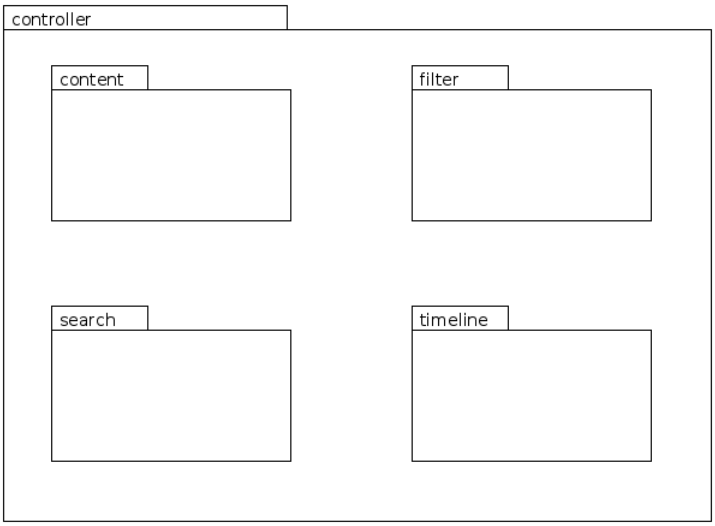

Figura 16: Diagramma del package *controller*

Questo capitolo illustra i componenti del *Controller*, per ciascuno dei quali è indicato il nome della classe accompagnata da un'identificazione sintetica, separati dal carattere '|' (separatore verticale).

5.1 PACKAGE CONTROLLER

- Avvio di una ricerca;
- Esecuzione di una ricerca (ambiti e motori di ricerca);
- Aggiornamento dei risultati di ricerca (modifica entità);
- Aggiornamento dei risultati di ricerca (aggiornamento filtri);
- Navigazione della timeline;
- Visualizzazione discussione.

5.1.1 *MainController* | *Gestore dell'applicazione*

La classe *MainController* gestisce l'inizializzazione delle componenti del sistema e il caricamento dell'interfaccia grafica.

5.1.2 *ContentController* | *Gestione dei contenuti informativi*

La classe *ContentController* gestisce le operazioni relative al reperimento e alla consultazione dei contenuti informativi, sia come risultati di una ricerca sia come elementi di una discussione.

5.1.3 *EntityController* | *Gestione delle entità*

La classe *EntityController* gestisce le operazioni relative alle entità del dominio individuate al termine della ricerca e con cui l'utente può interagire.

5.1.4 *FilterController* | *Gestione dei filtri*

La classe *FilterController* gestisce la creazione, il caricamento e l'interazione dell'utente con i filtri.

5.1.5 *FCList* | *Gestione dei filtri a lista di valori*

La classe *FCList* estende la classe *FilterController* e gestisce i filtri di tipo *FList*.

5.1.6 *FCRange* | *Gestione dei filtri ad intervallo di valori*

La classe *FCRange* estende la classe *FilterController* e gestisce i filtri di tipo *FRange*.

5.1.7 *FCSwitch* | *Gestione dei filtri ad interruttore*

La classe *FCSwitch* estende la classe *FilterController* e gestisce i filtri di tipo *FSwitch*.

5.1.8 *FCValue* | *Gestione dei filtri a soglia di valore*

La classe *FCValue* estende la classe *FilterController* e gestisce i filtri di tipo *FValue*.

5.1.9 *SearchController* | *Gestione della ricerca*

La classe *SearchController* gestisce le operazioni connesse alla ricerca di contenuti informativi nella piattaforma.

5.1.10 *TimelineController* | *Gestione della cronologia*

La classe *TimelineController* gestisce le operazioni riguardanti la navigazione temporale dei contenuti.

6 DESIGN PATTERN

Questo capitolo illustra i *design pattern* utilizzati nella progettazione, indicando a quali classi siano applicati.

6.1 ABSTRACT FACTORY

6.2 FACADE

- CriteriaModel;

6.3 SINGLETON

7 TRACCIAMENTO

Le informazioni relative al tracciamento componenti - requisiti sono disponibili nel file *tracciamento.ods*, allegato alla presente documentazione.