

PROGETTAZIONE

NICOLA MORETTO (MATR. 578258)

4 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 | Aggiornato il capitolo 3. |
| 2.2 | 30-10-2012 | Aggiornato il capitolo 4. |
| 2.3 | 02-11-2012 | Aggiornato il capitolo 5. |
| 2.4 | 03-11-2012 | Revisione integrale del documento. |
| 3.0 | 04-11-2012 | Pubblicazione della terza versione. |

Tabella 1: Registro delle modifiche

INDICE

| | | |
|--------|---------------------------|----|
| 1 | INTRODUZIONE | 8 |
| 1.1 | Convenzioni | 8 |
| 1.2 | Riferimenti informativi | 8 |
| 2 | ARCHITETTURA | 9 |
| 2.1 | Architettura generale | 9 |
| 2.2 | Componenti architetturali | 9 |
| 2.2.1 | Componente Model | 9 |
| 2.2.2 | Componente View | 9 |
| 2.2.3 | Componente Controller | 10 |
| 3 | COMPONENTE MODEL | 11 |
| 3.1 | Package model | 11 |
| 3.1.1 | Content | 11 |
| 3.1.2 | Metadata | 12 |
| 3.1.3 | Emotion | 12 |
| 3.1.4 | Intention | 12 |
| 3.1.5 | Interest | 12 |
| 3.1.6 | Topic | 13 |
| 3.1.7 | Type | 13 |
| 3.1.8 | User | 13 |
| 3.1.9 | Entity | 13 |
| 3.1.10 | EntityType | 13 |
| 3.1.11 | Label | 13 |
| 3.1.12 | Meaning | 13 |
| 3.2 | Package model.criteria | 13 |
| 3.2.1 | CriteriaModel | 13 |
| 3.2.2 | Criterion | 13 |
| 3.2.3 | CList | 14 |
| 3.2.4 | CRange | 14 |
| 3.2.5 | CValue | 14 |
| 3.3 | Package model.filter | 14 |
| 3.3.1 | FilterFactory | 15 |
| 3.3.2 | FilterModel | 15 |
| 3.3.3 | Filter | 16 |
| 3.3.4 | FList | 16 |
| 3.3.5 | FRange | 16 |
| 3.3.6 | FSwitch | 16 |
| 3.3.7 | FValue | 16 |
| 3.4 | Package model.provider | 16 |
| 3.4.1 | Provider | 17 |
| 3.4.2 | ProviderModel | 17 |
| 3.4.3 | ProviderRegistry | 17 |
| 3.4.4 | PLabel | 17 |
| 3.4.5 | PFullText | 18 |
| 3.5 | Package model.search | 18 |
| 3.5.1 | Search | 18 |
| 3.5.2 | EntityList | 18 |
| 3.5.3 | Query | 18 |

| | | |
|-------|---------------------------|----|
| 3.5.4 | Result | 19 |
| 3.5.5 | Scope | 19 |
| 3.5.6 | Term | 19 |
| 3.6 | Package model.timeline | 19 |
| 3.6.1 | Timeline | 19 |
| 3.6.2 | TimeUnit | 19 |
| 3.6.3 | Day | 20 |
| 3.6.4 | Week | 20 |
| 3.6.5 | Month | 20 |
| 3.6.6 | Year | 20 |
| 4 | COMPONENTE VIEW | 21 |
| 4.1 | Package view | 21 |
| 4.1.1 | MainWindow | 21 |
| 4.2 | Package view.content | 22 |
| 4.2.1 | ContentView | 22 |
| 4.2.2 | CVAnswer | 23 |
| 4.2.3 | CVEvent | 23 |
| 4.2.4 | CVMessage | 24 |
| 4.2.5 | CVQuestion | 24 |
| 4.2.6 | CVReview | 24 |
| 4.2.7 | CVTalk | 24 |
| 4.2.8 | CVThought | 24 |
| 4.2.9 | EntityView | 24 |
| 4.3 | Package view.filter | 24 |
| 4.3.1 | FilterContainer | 24 |
| 4.3.2 | FilterView | 25 |
| 4.3.3 | FVList | 25 |
| 4.3.4 | FVRange | 25 |
| 4.3.5 | FVSwitch | 25 |
| 4.3.6 | FVValue | 26 |
| 4.4 | Package view.search | 26 |
| 4.4.1 | SearchContainer | 26 |
| 4.4.2 | EntityListWidget | 26 |
| 4.4.3 | LabelEntityWidget | 27 |
| 4.4.4 | SearchBar | 27 |
| 4.4.5 | SearchScopeSelector | 27 |
| 4.5 | Package view.timeline | 27 |
| 4.5.1 | Timeline | 27 |
| 4.5.2 | TimeSlot | 28 |
| 4.5.3 | TimeAxis | 28 |
| 5 | COMPONENTE CONTROLLER | 29 |
| 5.1 | Package controller | 29 |
| 5.1.1 | MainController | 30 |
| 5.1.2 | ContentController | 30 |
| 5.1.3 | EntityController | 30 |
| 5.1.4 | TimelineController | 30 |
| 5.2 | Package controller.filter | 30 |
| 5.2.1 | FilterController | 30 |
| 5.2.2 | FCList | 30 |
| 5.2.3 | FCRange | 30 |
| 5.2.4 | FCSwitch | 31 |
| 5.2.5 | FCValue | 31 |

| | | |
|-------|---------------------------|----|
| 5.3 | Package controller.search | 31 |
| 5.3.1 | SearchController | 31 |
| 5.3.2 | ResultUpdate | 31 |
| 5.3.3 | RUListRefine | 31 |
| 5.3.4 | RURangeRefine | 32 |
| 5.3.5 | RUSwitchRefine | 32 |
| 5.3.6 | RUValueRefine | 32 |
| 5.3.7 | RULoadDiscussion | 32 |
| 5.3.8 | RULoadEntity | 32 |
| 6 | DESIGN PATTERN | 33 |
| 6.1 | Abstract Factory | 33 |
| 6.2 | Command | 33 |
| 6.3 | Decorator | 33 |
| 6.4 | Facade | 33 |
| 6.5 | Singleton | 33 |
| 7 | TRACCIAMENTO | 34 |

ELENCO DELLE FIGURE

| | | |
|-----------|---|----|
| Figura 1 | Diagramma del package <i>model</i> | 11 |
| Figura 2 | Diagramma delle classi del package <i>model</i> | 12 |
| Figura 3 | Diagramma delle classi del package <i>model.criteria</i> | 14 |
| Figura 4 | Diagramma delle classi del package <i>model.filter</i> | 15 |
| Figura 5 | Diagramma delle classi del package <i>model.provider</i> | 17 |
| Figura 6 | Diagramma delle classi del package <i>model.search</i> | 18 |
| Figura 7 | Diagramma delle classi del package <i>model.timeline</i> | 19 |
| Figura 8 | Diagramma del package <i>view</i> | 21 |
| Figura 9 | Diagramma delle classi del package <i>view</i> | 21 |
| Figura 10 | Diagramma delle classi del package <i>view.content</i> | 22 |
| Figura 11 | Informazioni essenziali di un contenuto | 22 |
| Figura 12 | Informazioni aggiuntive di un contenuto | 23 |
| Figura 13 | Diagramma delle classi del package <i>view.filter</i> | 25 |
| Figura 14 | Filtro ad intervallo di valori | 25 |
| Figura 15 | Filtro a doppio stato | 26 |
| Figura 16 | Filtro a soglia di valore | 26 |
| Figura 17 | Diagramma delle classi del package <i>view.search</i> | 26 |
| Figura 18 | Visualizzazione dei dettagli di un'entità | 27 |
| Figura 19 | Selezione dell'accezione delle etichette | 27 |
| Figura 20 | Diagramma delle classi del package <i>view.timeline</i> | 28 |
| Figura 21 | Asse temporale | 28 |
| Figura 22 | Diagramma del package <i>controller</i> | 29 |
| Figura 23 | Diagramma delle classi del package <i>controller</i> | 29 |
| Figura 24 | Diagramma delle classi del package <i>controller.filter</i> | 30 |
| Figura 25 | Diagramma delle classi del package <i>controller.search</i> | 31 |

INTRODUZIONE

1.1 CONVENZIONI

La nomenclatura adottata per i package e le classi è il *CamelCase*.

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).

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;
- ...

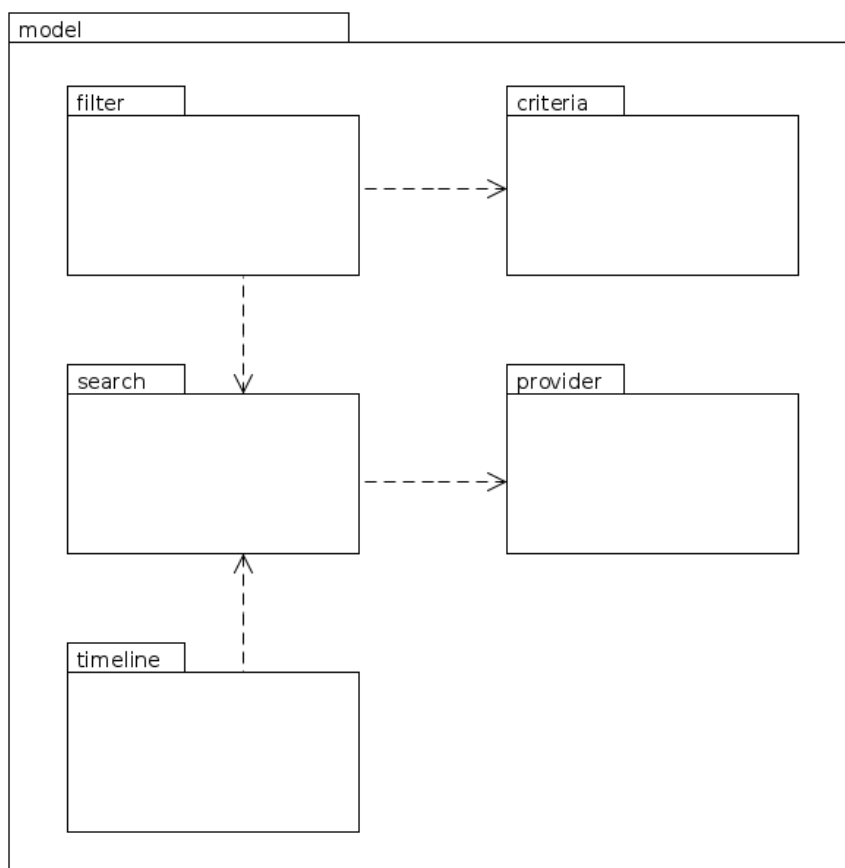


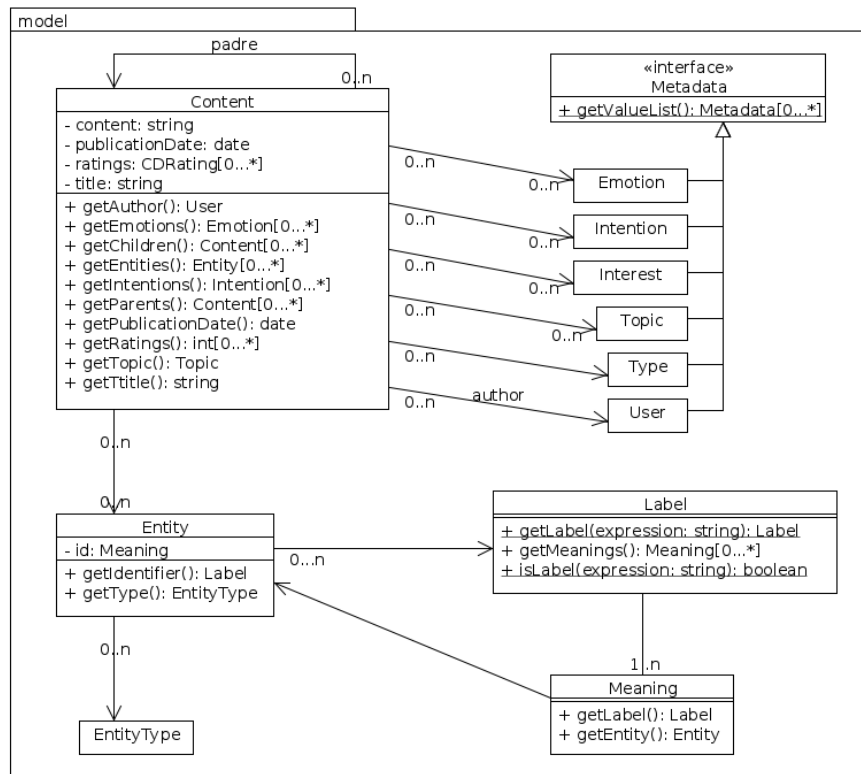
Figura 1: Diagramma del package *model*

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.

Figura 2: Diagramma delle classi del package *model*

3.1.2 *Metadata* | *Metadati dei contenuti*

L'interfaccia *Metadata* rappresenta una meta-informazione generica associata ad un contenuto informativo.

3.1.3 *Emotion* | *Emozioni*

La classe *Emotion* rappresenta un'emozione associabile ad un contenuto ed implementa l'interfaccia *Metadata*.

3.1.4 *Intention* | *Intenzione*

La classe *Intention* rappresenta un'intenzione associabile ad un contenuto ed implementa l'interfaccia *Metadata*.

3.1.5 *Interest* | *Interesse*

La classe *Interest* rappresenta gli interessi cui è associabile un contenuto ed implementa l'interfaccia *Metadata*.

3.1.6 *Topic* | *Argomento*

La classe *Topic* rappresenta un argomento cui può appartenere un contenuto ed implementa l'interfaccia *Metadata*.

3.1.7 *Type* | *Tipo di contenuto*

La classe *Intention* rappresenta un'intenzione associabile ad un contenuto ed implementa l'interfaccia *Metadata*.

3.1.8 *User* | *Autore*

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

3.1.9 *Entity* | *Entità del dominio*

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

3.1.11 *Label* | *Etichetta del dizionario*

La classe *Label* 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 *Meaning* rappresenta un'accezione di un'etichetta (*Label*) e si riferisce un'entità del dominio (*Entity*).

3.2 PACKAGE MODEL.CRITERIA

Ciascun contenuto può essere classificato in accordo a differenti parametri, 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 l'interfaccia standard dei criteri di classificazione.

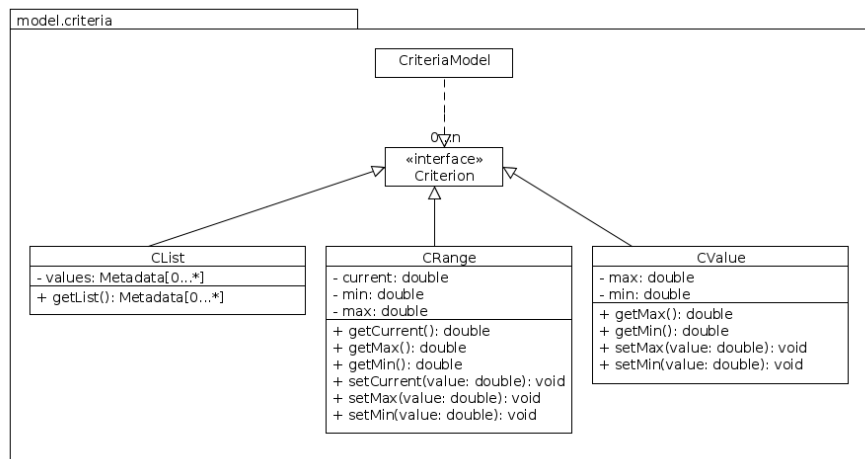


Figura 3: Diagramma delle classi del package *model.criteria*

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.

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

3.2.5 CValue | Filtro basato su una soglia di valore

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

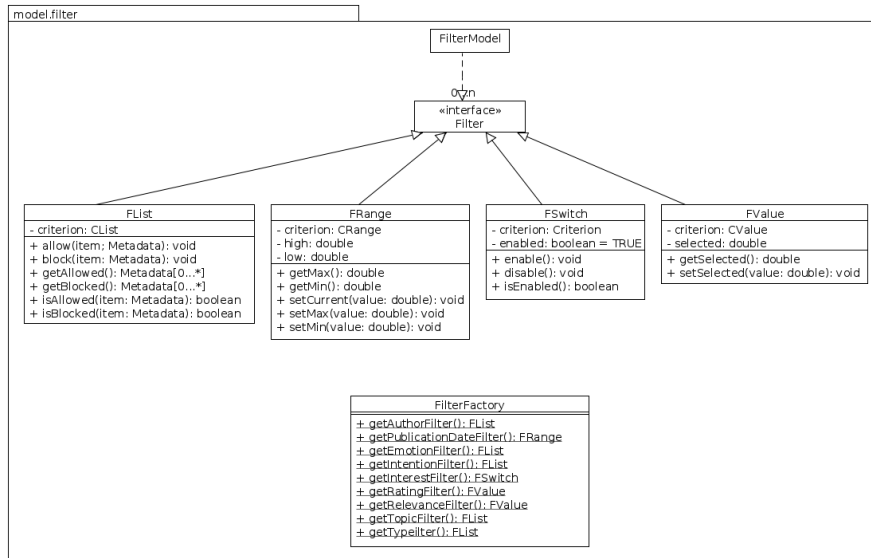
Ciascuna istanza viene utilizzata in combinazione con un filtro *FValue* o *FSwitch*, fornendo i valori e i parametri di configurazione iniziale.

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 |
|---------------------|-----------|---------|------------|
| <i>Argomento</i> | CList | FList | FList |
| <i>Attinenza</i> | CValue | FValue | FValue |
| <i>Autore</i> | CList | FList | FList |
| <i>Data di pub.</i> | CRange | FRange | FRange |
| <i>Emozioni</i> | CList | FList | FList |
| <i>Giudizi</i> | CValue | FValue | FValue |
| <i>Intenzioni</i> | CList | FList | FList |
| <i>Interessi</i> | CList | FSwitch | FSwitch |
| <i>Tipo</i> | CList | FList | FList |

Tabella 2: Dipendenze tra classi rilevanti per *FilterFactory*

3.3.2 *FilterModel* | Gestione dei filtri

Tale componente rappresenta l'interfaccia del package *model.filter*, utile a esporre le funzionalità per l'istanziamento 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 qualsiasi criterio (*Criterion*).

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.

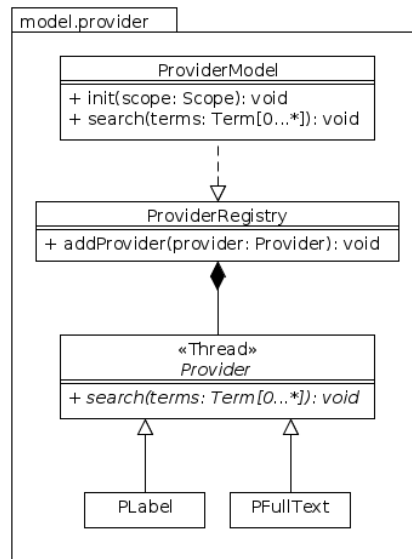


Figura 5: Diagramma delle classi del package *model.provider*

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*).

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

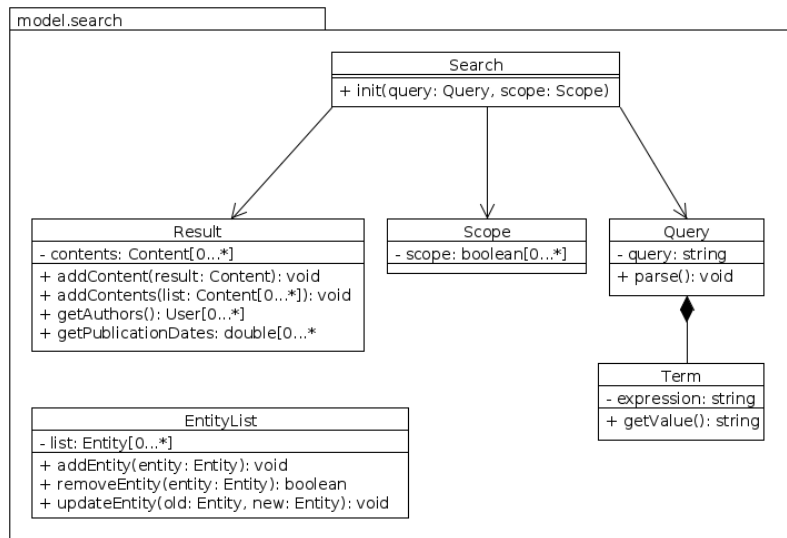


Figura 6: Diagramma delle classi del package *model.search*

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

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

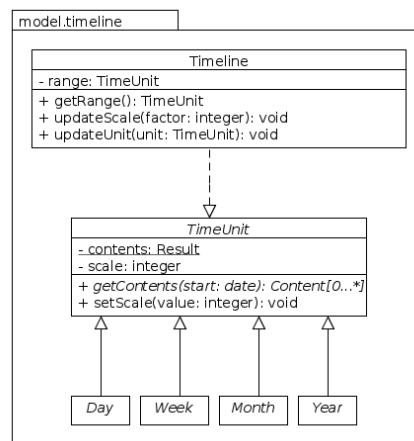


Figura 7: Diagramma delle classi del package *model.timeline*

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 (*TimeUnit*) di ampiezza fissa - personalizzabile dall'utente in termini di unità (giorni, settimane, mesi, anni) e scala (1, 2, ...) - in cui vengono collocati i risultati della ricerca o i contenuti di una discussione.

3.6.2 *TimeUnit* | Unità temporale

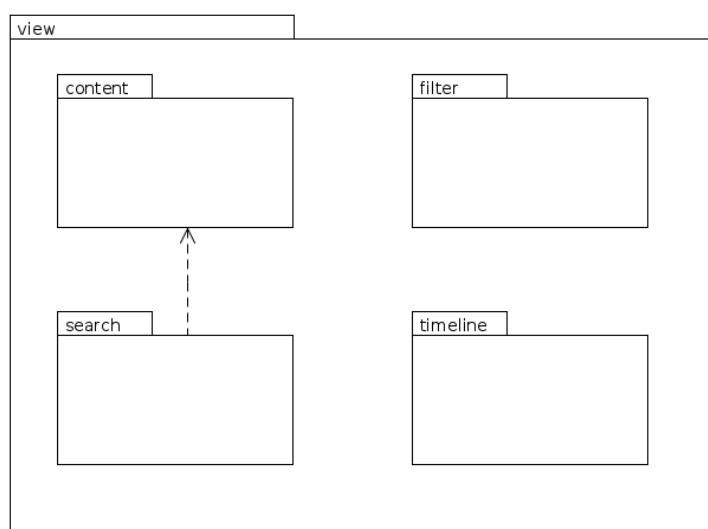
La classe *TimeUnit* rappresenta un generico intervallo di tempo, che consente di reperire i contenuti corrispondenti ai risultati di ricerca relativi ad una discussione e pubblicati nel periodo corrispondente.

3.6.3 *Day | Giorno (1-31)*

3.6.4 *Week | Settimana (1-52)*

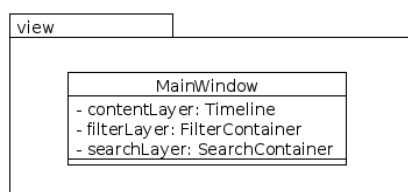
3.6.5 *Month | Mese (1-12)*

3.6.6 *Year | Anno*

Figura 8: 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).

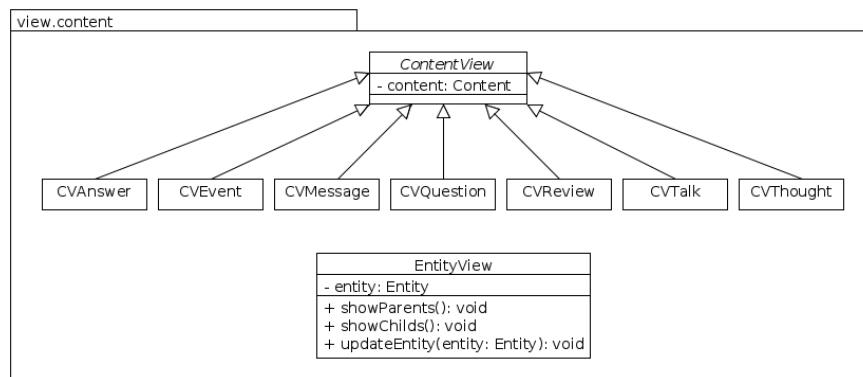
4.1 PACKAGE VIEW

Figura 9: Diagramma delle classi del 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

Figura 10: Diagramma delle classi del 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).

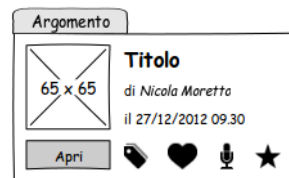


Figura 11: Informazioni essenziali di un contenuto

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

*Informazioni
essenziali*

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

ATTINENZA Il grado di attinenza di un contenuto rispetto ai criteri di ricerca 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 (*Type*) viene rappresentato graficamente variando la forma¹ dell'elemento grafico.

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

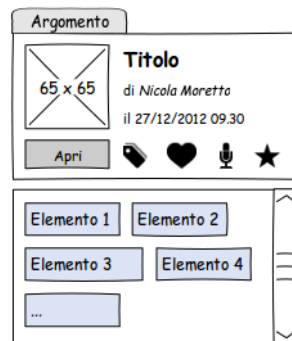


Figura 12: Informazioni aggiuntive di un contenuto

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

informazioni
aggiuntive

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

EMOZIONI A ciascun contenuto possono essere associate diverse emozioni (*Emotion*), 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.

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 associato ad un contenuto è un'informazione rappresentabile in formato testuale o grafico.

INTENZIONI A ciascun contenuto possono essere associate delle intenzioni (*Intention*), 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 *CVAnswer* rappresenta il componente grafico che visualizza un contenuto di tipo *risposta* ed estende la classe *ContentView*.

4.2.3 CVEvent | Evento

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

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, ...

4.2.4 *CVMessage* | Comunicazione privata

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

4.2.5 *CVQuestion* | Domanda

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

4.2.6 *CVReview* | Recensione

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

4.2.7 *CVTalk* | Discorso

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

4.2.8 *CVThought* | Pensiero

La classe *CVThought* 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:

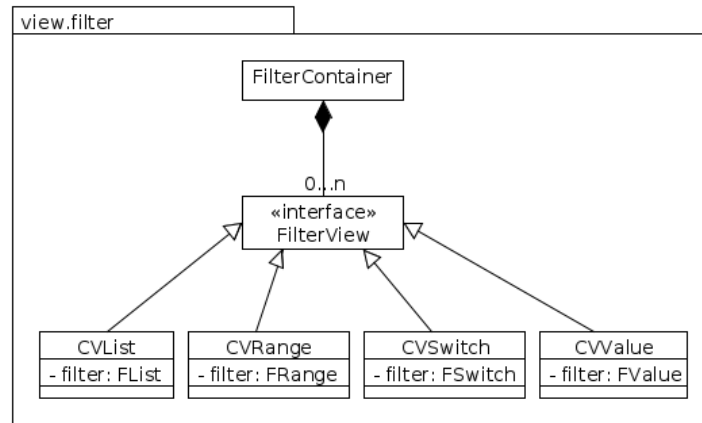
- visualizzare la lista delle entità padre;
- visualizzare la lista delle entità figlie;
- sostituire l'entità corrente con un padre o un figlio.
- eliminare l'entità corrente.

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.

Figura 13: Diagramma delle classi del package *view.filter*

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.

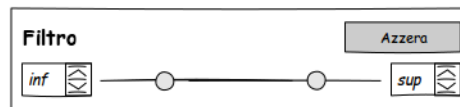


Figura 14: Filtro ad intervallo di valori

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 15: 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.

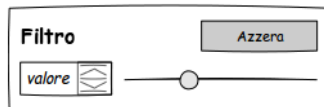
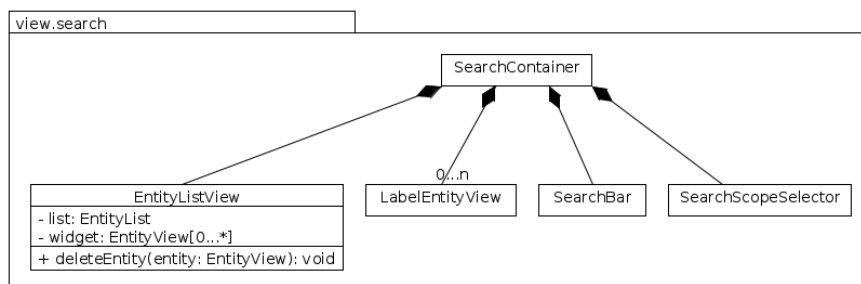


Figura 16: Filtro a soglia di valore

4.4 PACKAGE VIEW.SEARCH

Figura 17: Diagramma delle classi del 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.

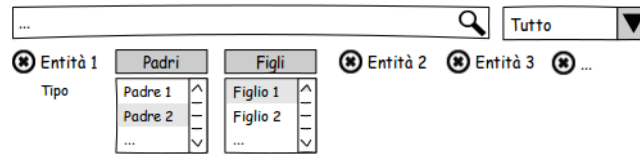


Figura 18: Visualizzazione dei dettagli di un'entità

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 19: 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.

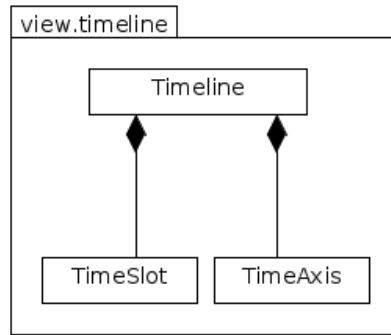


Figura 20: Diagramma delle classi del package *view.timeline*

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.

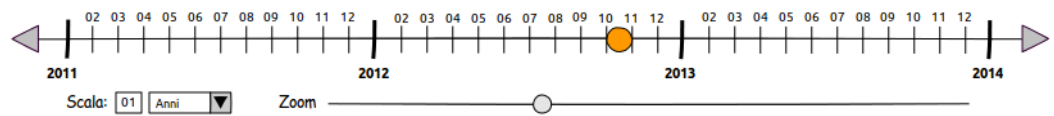
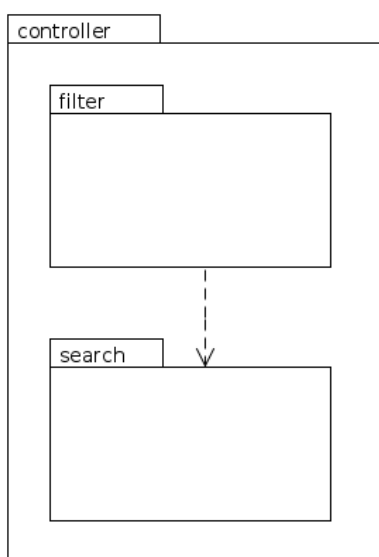


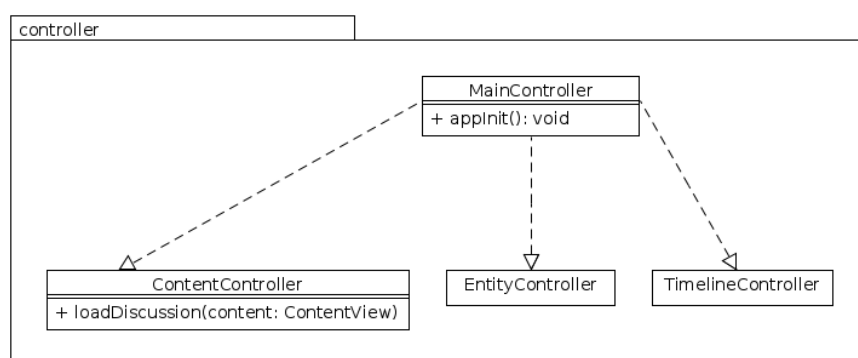
Figura 21: Asse temporale

COMPONENTE CONTROLLER

Figura 22: 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

Figura 23: Diagramma delle classi del package *controller*

5.1.1 *MainController* | Gestione 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 *TimelineController* | Gestione della cronologia

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

5.2 PACKAGE CONTROLLER.FILTER

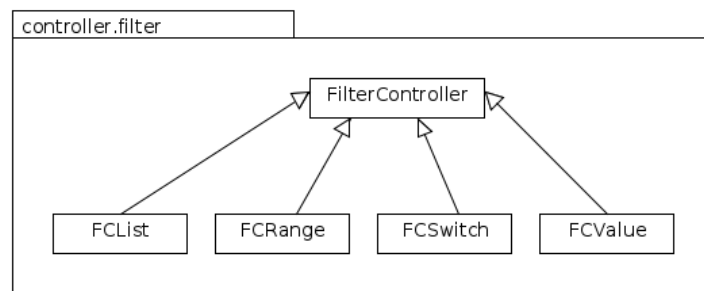


Figura 24: Diagramma delle classi del package *controller.filter*

5.2.1 *FilterController* | Gestione dei filtri

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

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

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

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

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

5.2.4 *FCSwitch* | Gestione dei filtri ad interruttore

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

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

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

5.3 PACKAGE CONTROLLER.SEARCH

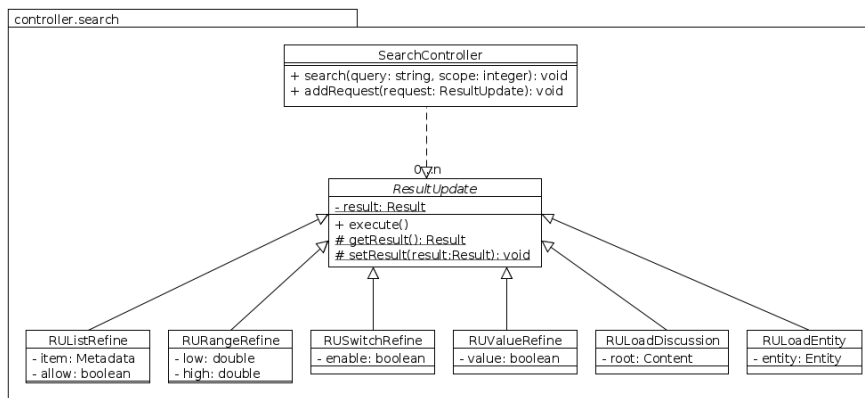


Figura 25: Diagramma delle classi del package *controller.search*

5.3.1 *SearchController* | Gestione della ricerca

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

5.3.2 *ResultUpdate* | Gestione dell'aggiornamento dei risultati di una ricerca

La classe *ResultUpdate* rappresenta l'interfaccia delle operazioni inerenti l'aggiornamento dei risultati di ricerca:

- alterazione dei filtri di ricerca;
- eliminazione di un'entità cercata;
- visualizzazione di una discussione (a partire da un risultato di ricerca).

5.3.3 *RURefine* | Gestione del raffinamento dei risultati di ricerca

La classe *RUListRefine* incapsula il comando di aggiornamento dei risultati di ricerca visualizzati a fronte dell'alterazione di un filtro *FList*.

5.3.4 *RURangeRefine* | Gestione del raffinamento dei risultati di ricerca

La classe *RURangeRefine* incapsula il comando di aggiornamento dei risultati di ricerca visualizzati a fronte dell'alterazione di un filtro *FRange*.

5.3.5 *RUSwitchRefine* | Gestione del raffinamento dei risultati di ricerca

La classe *RUSwitchRefine* incapsula il comando di aggiornamento dei risultati di ricerca visualizzati a fronte dell'abilitazione o disattivazione di un filtro *FSwitch*.

5.3.6 *RUValueRefine* | Gestione del raffinamento dei risultati di ricerca

La classe *RUValueRefine* incapsula il comando di aggiornamento dei risultati di ricerca visualizzati a fronte dell'alterazione di un filtro *FValue*.

5.3.7 *RULoadDiscussion* | Gestione del raffinamento dei risultati di ricerca

La classe *RULoadDiscussion* incapsula il comando di aggiornamento dei risultati di ricerca a fronte della visualizzazione della discussione generata da un contenuto informativo visualizzato.

5.3.8 *RULoadEntity* | Gestione del raffinamento dei risultati di ricerca

La classe *RULoadEntity* incapsula il comando di aggiornamento dei risultati di ricerca a fronte dell'eliminazione di un'entità.

DESIGN PATTERN

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

6.1 ABSTRACT FACTORY

- FilterFactory

6.2 COMMAND

- ResultUpdate
 - RUListRefine
 - RURangeRefine
 - RUSwitchRefine
 - RUValueRefine
 - RULoadDiscussion
 - RULoadEntity

6.3 DECORATOR

- FSwitch

6.4 FACADE

- CriteriaModel
- Filter
- ProviderModel
- Search
- FilterContainer
- SearchContainer
- Timeline

6.5 SINGLETON

- MainController

TRACCIAMENTO

Le informazioni relative al tracciamento - requisiti verranno rese disponibili successivamente in un file *tracciamento.ods*.