CloudStore
Relazione di progetto
Basi Di Dati

Mazzoli Alessandro

21 giugno 2021

Indice

Capitolo 1

Analisi dei requisiti

1.1 Intervista

Si vuole sviluppare un applicativo per il cloud storage tramite il quale gli utenti potranno fare l'upload dei propri file in modo da salvarli in cloud ed averne sempre l'accesso tramite la piattaforma web. Sarà inoltre possibile la condivisione dei propri documenti tra utenti.

1.1.1 Clienti

Gli utenti una volta registrati sulla piattaforma ne avranno accesso tramite le proprie credenziali che consistono nell'indirizzo email che identifica univocamente un utente e la relativa password.
Si vogliono inoltre salvare i dati anagrafici dei clienti quali codice fiscale, nome, cognome, numero di telefono, data di nascita e residenza. Questi dati verranno utilizzati per contattare l'utente nel caso ci siano dei dubbi sulla legalità del materiale che l'utente salva in cloud o in caso di qualsiasi problematica concernente il suo account.
Inoltre le policy della piattaforma richiedono che gli utenti abbiano almeno 18 anni per potersi iscrivere.

1.1.2 Gestione File

I file saranno gestiti tramite un file system gerarchico basato sulle cartelle proprio come quello dei normali pc.
Avremo quindi che un utente una volta essere entrato nella piattaforma si ritroverà nella propria "Home Folder" dove potrà creare nuove cartelle oppure caricare, spostare, copiare, o eliminare i propri file.
Ogni file sarà identificato relativamente alla cartella in cui si trova tramite il proprio nome e la propria estensione.
Si vogliono inoltre gestire le versioni dei file, ovvero si potrà caricare una versione più nuova di un file già presente e di quest'ultimo verrà comunque tenuta traccia di tutte le versioni precedenti nel caso un giorno se ne voglia visualizzare o recuperare una in particolare.

Di una versione si vuole sapere il numero progressivo che la identifica rispetto al file a cui fa riferimento, la data di caricamento e la sua dimensione, che è data da quella dell'ultima versione.
Chiunque abbia accesso ad un file come già detto in precedenza potrà visionarlo in qualsiasi momento e potrà anche quindi scaricarlo nella macchina da cui è connesso. Sia delle visualizzazioni che dei download si vuole tenere traccia in modo da sapere chi ha visualizzato o scaricato un determinato file, salvando quale è stata la versione visionata/scaricata e quando è stata fatta l'operazione,
Si potranno inoltre aggiungere i file alla sezione dei "Preferiti" in modo da ritrovarli più facilmente.

1.1.3 Condivisione File

I file dei vari utenti potranno essere condivisi dando i permessi ad un account ad un determinato file, i quali si basano sui due criteri di lettura e scrittura. Andando quindi a lavorare su un singolo file sarà possibile condividerlo con qualsiasi numero di persone si voglia specificando per ognuno di queste quali permessi ha su codesto file.

1.1.4 Assistenza Clienti

Come già detto in precedenza la piattaforma offrirà un servizio di assistenza clienti tramite delle segnalazioni.
Il personale incaricato all'assistenza sono gli operatori certificati CloudStore, ognuno dei quali ha un proprio codice identifico e una propria password di accesso al sistema di assistenza. Si vuole inoltre tenere traccia dell'anagrafica del personale.
Il cliente nel caso riscontrasse un qualsiasi problema non dovrà fare altro che andare nell'apposita sezione di assistenza della piattaforma e aprire una segnalazione. Nell'apertura dovrà specificare quale sia il problema che ha riscontrato descrivendolo al meglio, in modo da semplificare e velocizzare la risoluzione del problema.
La segnalazione verrà notificata agli operatori con meno carico di lavoro, i quali potranno decidere se accettarlo o meno. Una volta che una segnalazione è stato accettata da un operatore non sarà più accettabile dagli altri, sarà quindi compito dell'operatore che la ha accettata portare a termine la risoluzione del problema.
Poiché non sempre è tutto chiaro dal messaggio di segnalazione del cliente l'operatore potrà fare delle ulteriori domande sulla segnalazione tramite degli interventi che corrispondono allo scambio di messaggi tra utente e operatore.
Una volta risolta la problematica l'operatore potrà procedere alla chiusura della segnalazione.

1.2 Definizione delle specifiche in linguaggio naturale ed estrazione dei concetti principali

1.2.1 Glossario Termini

Termine	Descrizione	Sinonimi
Utente	La persona iscritta sulla piattaforma che può caricare/scaricare/modificare i suoi file	Account, Cliente
File	Il file che viene caricato sulla piattaforma dall'utente	Documento
Directory	Una collezione di file e altre cartelle, ogni utente ne ha una principale e ne può creare altre ricorsivamente	Cartella, Folder
Versione	Una singola istanza del file in un determinato istante di tempo. Ogni versione di un file è specifica di una determinata data di caricamento	Istanza di file
Download	L'operazione di salvataggio in locale da parte di un utente	Scaricamento
Operatore	Un operatore dell'azienda CloudStore il cui compiti è risolvere i problemi riscontrati dagli utenti della piattaforma	Assistente
Condivisione	Fornire accesso ad un determinato file ad un utente che non ne è il proprietario, specificando quali permessi ha tra lettura e scrittura	Sharing
Segnalazione	Richiesta di aiuto per la risoluzione di una problematica da parte di un cliente. Un operatore provvederà ad accettare la segnalazione e dopo averla risolta la chiuderà	Richiesta di aiuto
Intervento	Ogni segnalazione prevede una serie di interventi che consistono in uno scambio di messaggi tra cliente e operatore	Messaggio

1.2.2 Concetti Principali

Una volta riassunti quali sono i termini principali e di conseguenze le entità che sono in gioco andiamo ad analizzarle.
L'**Utente** corrisponde appunto all'account sulla piattaforma, di esso ci occorre tenere traccia dell'email che sarà il suo identificativo per accedere alla piattaforma e della relativa password.
Ci servirà inoltre sapere dati personali quali nome, cognome, data di nascita (poiché sono accettati solo persone con più di 16 anni). Si decide di tenere traccia anche della data di registrazione dell'account per fini statistici.
I **File** sono i documenti caricati dagli utenti sulla piattaforma, sono identificati tramite il loro nome e la loro estensione ma solo relativamente alla cartella in cui si trovano, infatti possono esistere file con stessi nomi ed estensioni ma solo in cartelle diverse. Si deve inoltre sapere la data di creazione e la dimensione, caratteristica che però è della versione, in particolare dell'ultima a cui fa riferimento.
Le **Directory** sono le cartelle che contengono i file, ma possono a loro volta contenere altre directory ricorsivamente. Anch'esse sono identificate da un identificativo univoco e si vuole sapere qual'è il proprio nome, la data di creazione.
Come risulta dall'intervista l'applicativo deve tenere traccia delle **Versioni** dei file che vengono aggiornate nel tempo. Ognuna di queste è identificata da un codice progressivo relativo al file a cui fa riferimento e si vuole tenere traccia della data di creazione, della dimensione e del link al file effettivo che viene hostato nel data-center di CloudStore.
Poiché si vuole tenere traccia anche dei **Download** di essi ci servirà sapere quando sono stati effettuati. Anch'essi saranno identificati tramite un codice identificativo numerico.
Per quanto riguarda le **Condivisioni** esse sono identificate dalla coppia File-Utente che specificherà inoltre quali permessi ha tra lettura e scrittura. Sempre per scopi di analisi si terrà inoltre traccia di quando è stato concesso il permesso.
Passando alla parte che riguarda l'assistenza clienti sulla piattaforma troviamo l'**Operatore**, ovvero l'impiegato dell'azienda che si occupa di risolvere le problematiche. Ogni operatore ha un proprio codice identificativo numerico e si vuole tenere traccia dei suoi dati anagrafici quali codice fiscale, nome, cognome, data di nascita e città di residenza.
Poiché l'operatore dovrà avere alla sezione della piattaforma riservata agli operatori bisognerà tenere traccia della sua password.
La **Segnalazione** consiste appunto nella notifica da parte di un utente che ha riscontrato un problema e decide di contattare l'assistenza. Nella segnalazione dovrà essere presente una descrizione del problema riscontrato in modo che l'operatore possa già capire se è di sua competenza o meno. Quando un operatore avrà accettato la segnalazione avverrà uno scambio di messaggi che prendono il nome di **Interventi** tra l'utente e l'operatore con lo scopo di risolvere la problematica. Di questi interventi si vuole appunto tenere traccia del contenuto dei messaggi e della data in cui sono stati inviati. Ogni intervento ha un codice progressivo rispetto alla segnalazione a cui si riferisce, in modo da poter stabilire l'ordine esatto dei messaggi anche senza l'utilizzo del confronto delle date.

Capitolo 2

Progettazione Concettuale

2.1 Schema scheletro

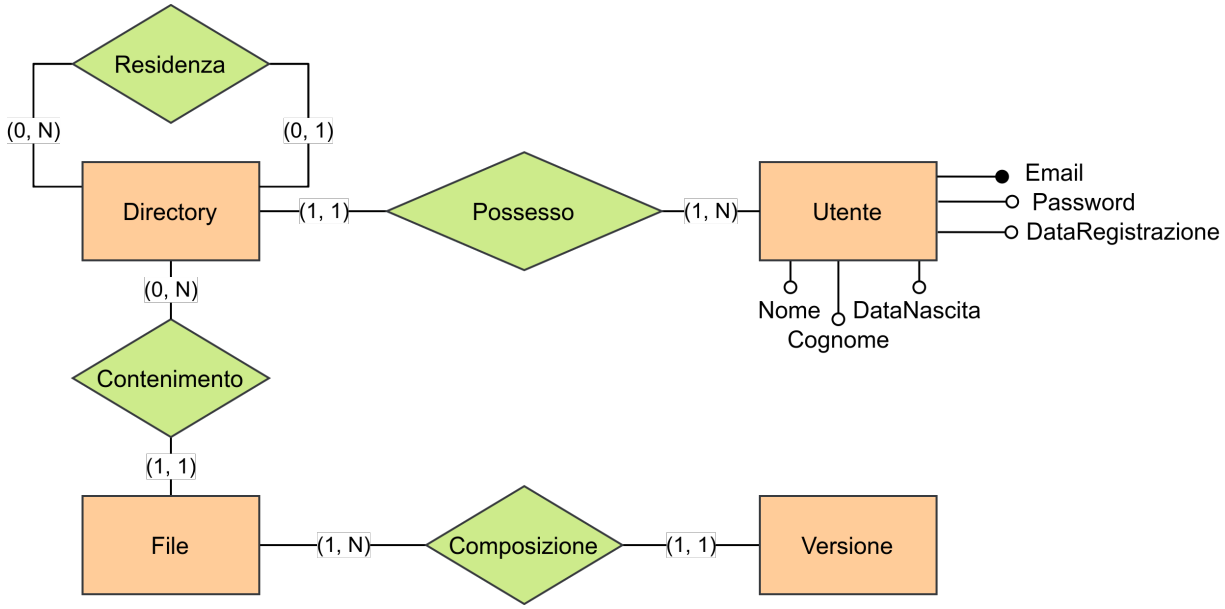

Figura 2.1: Schema scheletro del fulcro del dominio

2.2 Raffinamenti proposti

2.2.1 File System

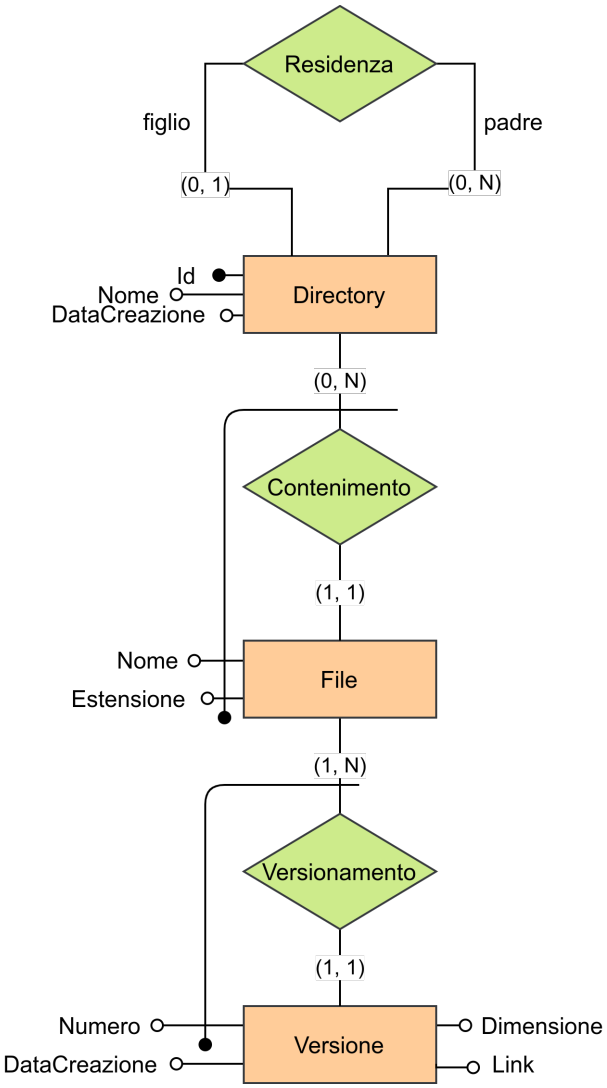

Figura 2.2: Schema gestione file system

Come già accennato in precedenza il salvataggio dei file dell'utente verrà gestito tramite un comune file system gerarchico che però tiene anche traccia delle versioni dei file.
Sarà quindi necessario fare in modo di avere la struttura di directory e file in modo che una directory possa contenere più file ma anche che una directory possa contenere anche altre cartelle al suo interno in modo ricorsivo.
Con lo schema precedentemente mostrato abbiamo quindi raggiunto il nostro scopo, infatti

come possiamo vedere una Directory è una sottocartella di 0 o 1 directory, 0 nel caso sia la radice dell'albero del file system e 1 nel caso sia una sottocartella.
Di seguito abbiamo la gestione dei file, dove ognuno di essere è contenuto in una e una sola cartella.
Per la gestione delle versioni si è appunto decide si utilizzare l'entità versione che viene identificata con numero progressivo relativo al file a cui si riferisce e che contiene il link al file effettivo nel data center. In questo modo le versioni verranno numerate progressivamente e sarà semplice la gestione di quest'ultime.

2.2.2 Condivisione & Preferiti

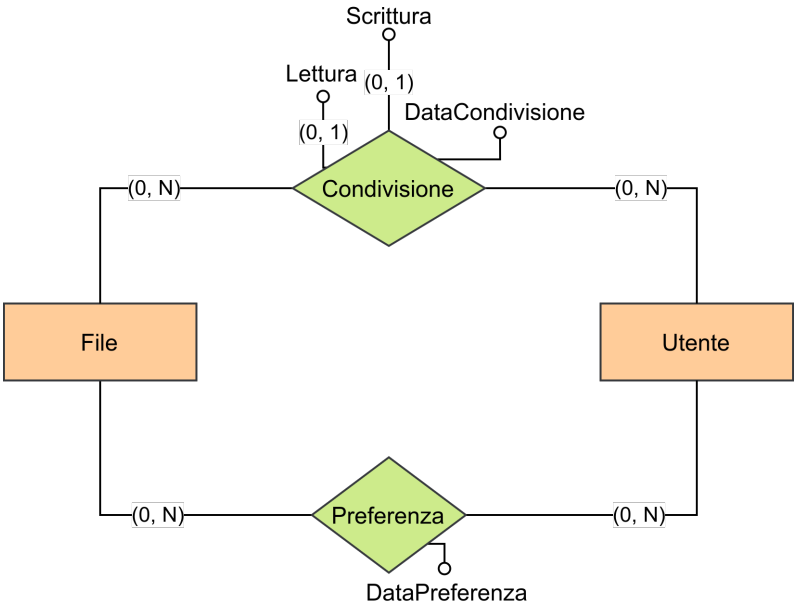

Figura 2.3: Schema gestione condivisioni

Per quanto concerne la gestione delle condivisioni dei file si è deciso di utilizzare la relazione Condivide, che identifica univocamente per ogni coppia File - Utente quali sono i permessi che quell'utente ha su quel determinato file.
Si è deciso di tenere traccia anche della data in cui è stato concesso questo permesso in modo da potere fare delle analisi sul sistema dopo avere raggiunto un numero abbastanza grande di utenti.

Per l'aggiunta di specifici file alla sezione dei preferiti si è utilizzata la relazione Preferisce che collega uno specifico file ad un utente. Anche in questo caso è interessante tenere traccia di quando il file è stato aggiunto ai preferiti.

2.2.3 Segnalazioni

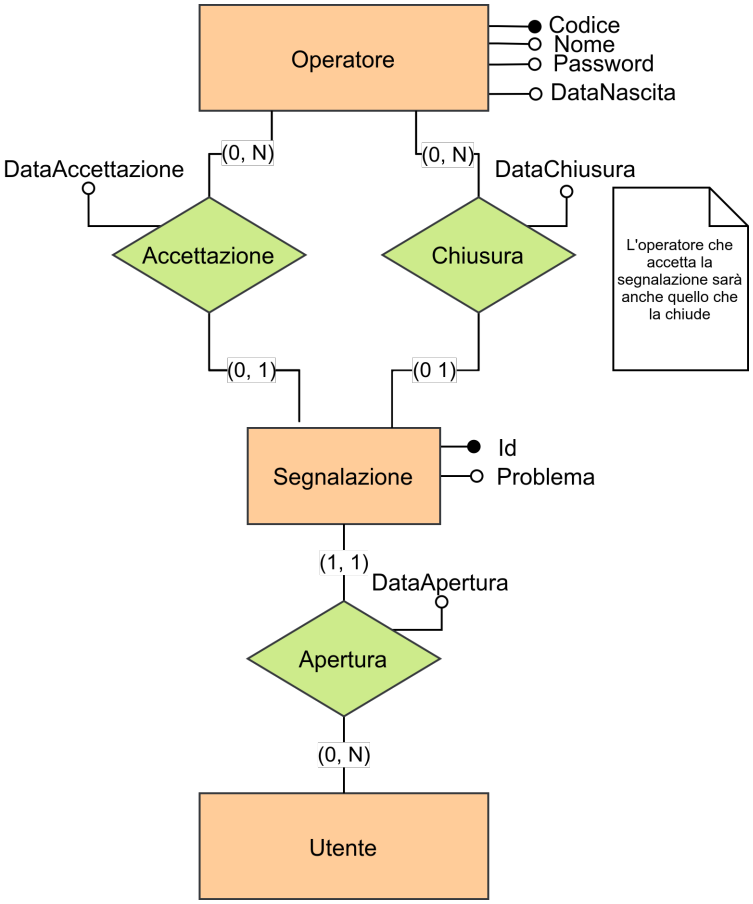

Figura 2.4: Schema gestione segnalazioni

La gestione delle segnalazioni al servizio di assistenze deve fare in modo di mettere in contatto l'utente con il personale adibito.
Abbiamo quindi la relazione di apertura che collega l'utente alla segnalazione che ha deciso di aprire salvandone in questo casa la data di apertura.
Avremo quindi poi l'operatore che tramite la relazione Accetta potrà prendere nelle sue mani la gestione della segnalazione e una volta risolta potrà procedere alla chiusura di quest'ultima.
Come si può notare dalla nota non è previsto che l'operatore che chiude la segnalazione sia diverso da quello che la ha accettata, sarà quindi necessario trovare successivamente una soluzione a questa ambiguità.

2.2.4 Interventi

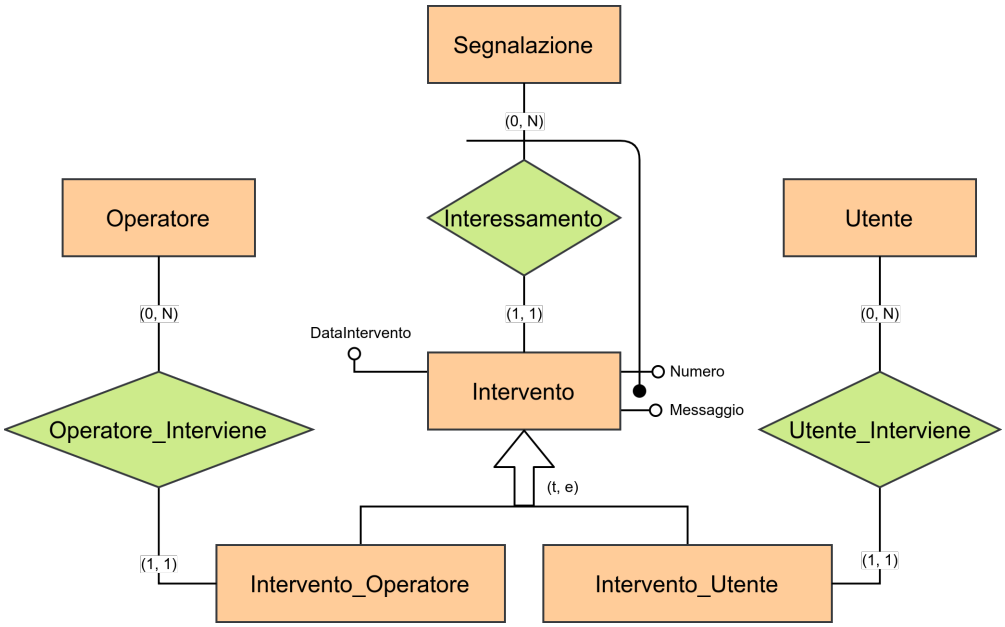

Figura 2.5: Schema gestione interventi

Come già anticipato dovrà essere possibile uno scambio di messaggi tra l'operatore e l'utente.
Questo è ottenuto tramite l'entità intervento che viene identificato con un numero progressivo relativo alla segnalazione a cui si riferisce e che tiene traccia del messaggio che si sta inviando e della data in cui è stato inviato.
Poiché un intervento può essere fatto sia da un operatore che da un utente si è giunti all'utilizzo di una gerarchia che definisce i due tipi di interventi possibili per una segnalazione: quello da parte dell'operatore e quello da parte dell'utente.

2.2.5 Tracciamento

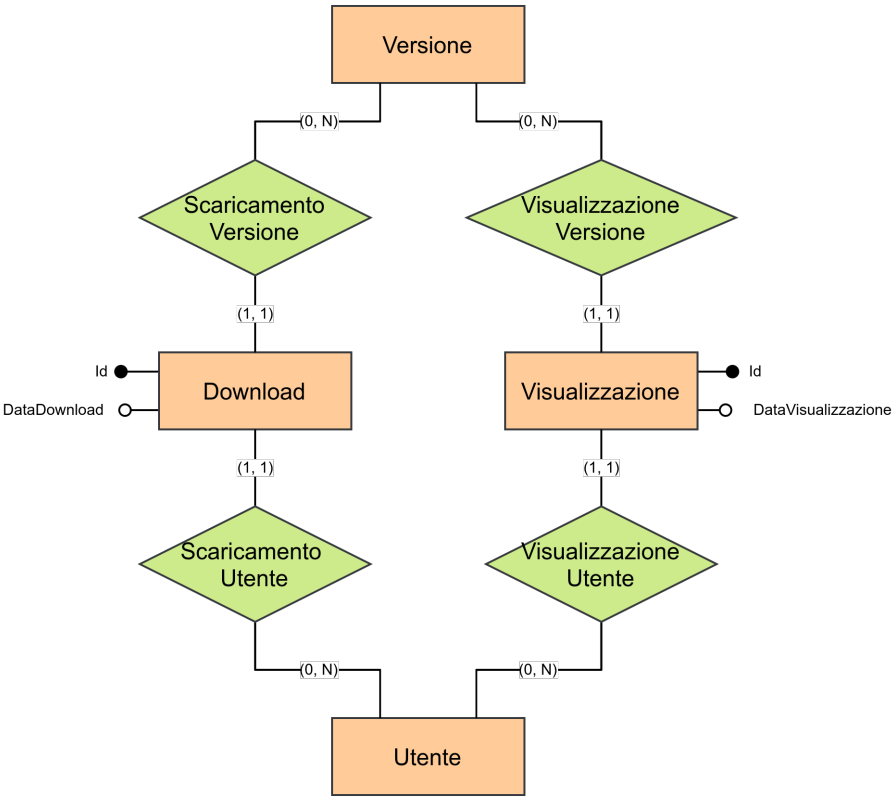

Figura 2.6: Schema gestione tracciamento

Il tracciamento delle attività dell'utente quali lo scaricamento o la visualizzazione di file è stato ottenuto tramite la gestione esplicitata nelle schema sopra riportato.
Essi non sono direttamente collegati ad un file perché effettivamente quella che noi visualizziamo o scarichiamo è una versione dello specifico file.
Quindi l'entità Download collegherà una versione di un file ad un determinato utente specificando quando è stato scaricata la specifica versione.
Non si è potuto optare per la una relazione Download poiché questa avrebbe reso impossibile per un utente scaricare più volte la stessa versione di un determinato file.

2.3 Schema concettuale finale

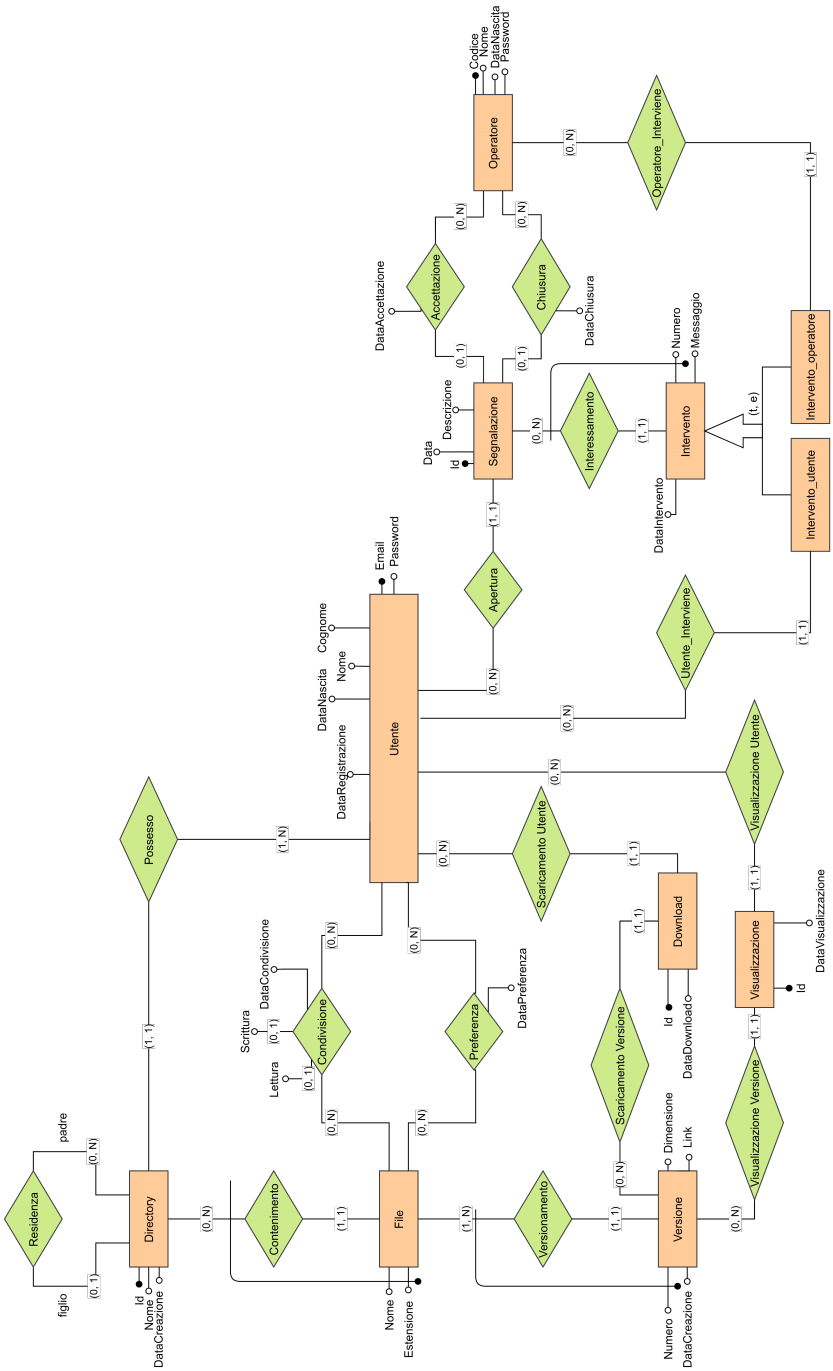

Figura 2.7: Schema ER finale

Capitolo 3

Progettazione logica

3.1 Stima del volume dei dati

Concetto	**Costrutto**	**Volume**
Utente	E	10000
Directory	E	200000
File	E	1000000
Versione	E	3000000
Download	E	2000000
Visualizzazione	E	10000000
Segnalazione	E	1000
Operatore	E	200
Intervento	E	5000
Condivisione	R	750000
Preferenza	R	50000
Residenza	R	190000
Contenimento	R	1000000
Versionamento	R	3000000
Scaricamento Versione	R	2000000
Scaricamento Utente	R	2000000
Visualizzazione Versione	R	10000000
Visualizzazione Utente	R	10000000
Possesso	R	200000
Apertura	R	1000
Accettazione	R	1000
Chiusura	R	1000
Utente Interviene	R	2500
Operatore Interviene	R	2500
Interessamento	R	5000

3.2 Descrizione delle operazioni principali e stima della loro frequenza

3.2.1 Operazioni Standard

Questa sezione è dedicata alle operazioni più basilari che consistono nell'inserimento o nella modifica di dati nel database.
La maggior parte di queste operazioni sono fatte dal software in automatico in base all'interazione dell'utente o dell'operatore con la piattaforma.

Codice	Operazione	Frequenza
1	Registrare un nuovo utente	5 volte al giorno
2	Creare una nuova directory	5000 volte al giorno
3	Aggiungere un nuovo file	10000 volte al giorno
4	Rinominare un file	5000 volte al giorno
5	Aggiungere una nuova versione di un file	5000 volte al giorno
6	Scaricare una versione di un file	1000 volte al giorno
7	Aggiungere un file ai preferiti	100 volte al giorno
8	Condividere un file con un utente	100 volte al giorno
9	Visualizzare un file	1000 volte al giorno
10	Aprire una segnalazione	10 volte al giorno
11	Accettare una segnalazione	10 volte al giorno
12	Chiudere una segnalazione	10 volte al giorno
13	Fare un intervento da parte di un utente	30 volte al giorno
14	Fare un intervento da parte di un operatore	30 volte al giorno
15	Registrare un nuovo operatore	10 volte all'anno

3.2.2 Analisi CloudStore

La sezione è dedicata alle operazioni di analisi dei dati da parte dei data scientists di CloudStore che giornalmente fanno delle operazioni di natura statistica e qualitativa sul sistema.

Codice	Operazione	Frequenza
16	Visualizzare l'operatore che ha chiuso più interventi	1 volta al giorno
17	Visualizzare il tipo di file più presente	1 volta al giorno
18	Visualizzare il numero di file preferiti per utente	1 volta al giorno
19	Visualizzare il numero di cartelle per utente	1 volta al giorno
20	Visualizzare il file più condiviso	1 volta al giorno
21	Visualizzare il file più visualizzato	1 volta al giorno
22	Visualizzare l'utente che ha aperto più segnalazioni	1 volta al giorno
23	Visualizzare il file più scaricato	1 volta al giorno

3.2.3 Operazioni Utenti

Le seguenti operazioni riguardano invece le azioni più frequenti da parte degli utenti sulla piattaforma.

Codice	Operazione	Frequenza
24	Visualizzare tutti i file scaricati di un utente	10 volte al giorno
25	Visualizzare per un file il numero di versioni	10000 volte al giorno
26	Visualizzare per un utente il numero di file	1000 volte al giorno
27	Visualizzare per un utente il numero di directory	1000 volte al giorno
28	Visualizzare per un utente il numero di file preferiti	1000 volte al giorno
29	Visualizzare il numero di file in una cartella	10000 volte al giorno
30	Visualizzare i file in una cartella	40000 volte al giorno
31	Visualizzare il numero di cartelle in una cartella	2000 volte al giorno
32	Visualizzare le cartelle in una directory	40000 volte al giorno
33	Visualizzare la dimensione di un file	20000 volte al giorno
34	Visualizzare la dimensione di una cartella	5000 volte al giorno

3.3 Schemi di navigazione e tabelle degli accessi

Di seguito verranno descritte solamente le operazioni più interessanti tra quelle descritte. Tutte le analisi dei costi delle successive operazioni saranno su operazioni che la cui analisi delle ridondanze non è stata fatta poiché ritenuta irrilevante per queste operazioni. Le operazioni con ridondanza sono riportate nel capitolo apposito successivamente.

3.3.1 01 - Registrare un nuovo utente

Concetto	Costrutto	Accesso	Tipo
Utente	E	1	S
Directory	E	1	S
Possesso	R	1	S
Totale	(3S) * 5 = 30 al giorno		

3.3.2 04 - Rinominare un file

Concetto	Costrutto	Accesso	Tipo
File	E	1	L
File	E	1	S
Totale	(1S + 1L)* 5000 = 15000 al giorno		

3.3.3 07 - Aggiungere un file ai preferiti

Concetto	Costrutto	Accesso	Tipo
Preferenza	R	1	S
Totale	(1S)* 100 = 200 al giorno		

3.3.4 11 - Accettare una segnalazione

Concetto	Costrutto	Accesso	Tipo
Accettazione	R	1	S
Totale	(1S)* 10 = 20 al giorno		

3.3.5 21 - Visualizzare il file più visualizzato

Concetto	Costrutto	Accesso	Tipo
File	E	1.000.000	L
Versionamento	R	3.000.000	L
Versione	E	3.000.000	L
Visualizzazione Versione	R	10.000.000	L
Totale	(17.000.000L)* 1 = 17.000.000 al giorno		

3.3.6 23 - Visualizzare il file più scaricato

Concetto	Costrutto	Accesso	Tipo
File	E	1.000.000	L
Versionamento	R	3.000.000	L
Versione	E	3.000.000	L
Scaricamento Versione	R	2.000.000	L
Totale	(9.000.000L)* 1 = 9.000.000 al giorno		

3.3.7 24 - Visualizzare tutti i file scaricati di un utente

Concetto	Costrutto	Accesso	Tipo
Utente	E	1	L
Scaricamento Utente	R	200	L
Download	E	200	L
Scaricamento Versione	R	200	L
Versione	E	200	L
Versionamento	R	200	L
File	E	200	L
Totale	(1201L)* 10 = 12010 al giorno		

3.3.8 28 - Visualizzare per un utente il numero di file preferiti

Concetto	Costrutto	Accesso	Tipo
Utente	E	1	L
Preferenza	R	5	L
Totale	(6L)* 1000 = 6000 al giorno		

3.3.9 30 - Visualizzare i file in una cartella

Concetto	Costrutto	Accesso	Tipo
Directory	E	1	L
Contenimento	R	5	L
File	E	5	L
Totale	(11L)* 40000 = 440000 al giorno		

3.4 Raffinamento dello schema

3.4.1 Gestione presa in carico degli operatori

Per la presa in carico delle segnalazioni poiché con le due relazioni **Accetta** e **Chiude** ci sarebbe il problema del controllare che l'operatore che chiude la segnalazione deve essere anche quello che la apre si è deciso di ristrutturare la situazione mettendo dentro **Segnalazione** l'id dell'operatore che prenderà in carico la segnalazione, mettendolo opzionale, in modo che venga settato una volta che un operatore abbia deciso di accettare la segnalazione.
Inoltre andranno anche aggiunti i due campi 'data_accettazione' e 'data_chiusura' in modo che la prima verrà valorizzata insieme all'identificatore dell'operatore che ha accettato la segnalazione, mentre la seconda verrà valorizzata a chiusura della segnalazione.

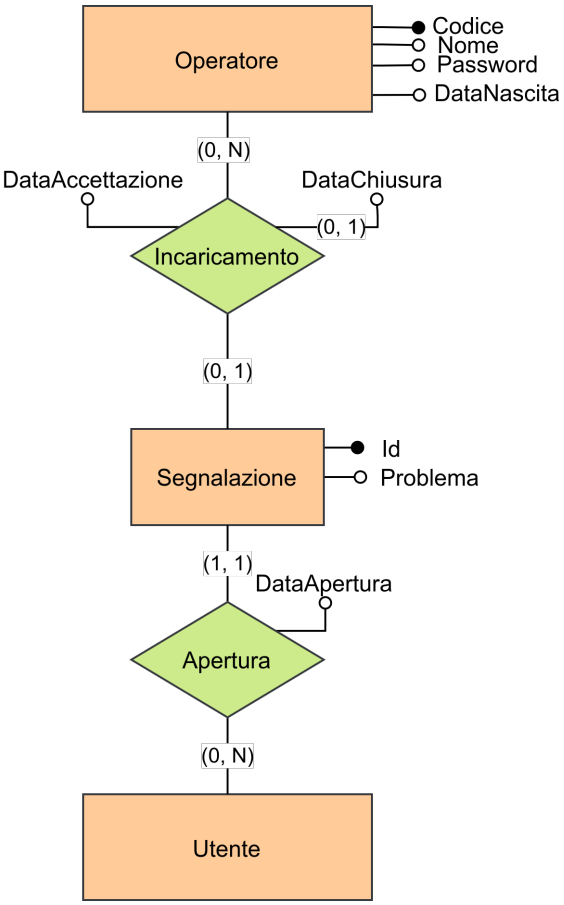

Figura 3.1: Raffinamento accettazione

3.4.2 Gerarchia degli interventi

Per la gerarchia dei due tipi di interventi (Intervento Utente, Intervento Operatore) si è deciso di optare per un collasso verso l'alto, portando dentro **Intervento** sia l'email dell'utente che ha fatto l'intervento sia quello dell'operatore.
In questo modo andrà aggiunta una restrizione che non permette a questi due campi di essere valorizzati contemporaneamente.
Cosi facendo possiamo sempre risalire a chi ha fatto l'intervento e capire se è stato un utente od un operatore.

3.4.3 Eliminazione degli identificatori esterni

Dallo schema principale sono state rimosse le seguenti relazioni:

- **Residenza** importando l'id della cartella padre dentro l'entità **Directory** e mettendolo opzionale
- **Contenimento** importando l'id della directory dentro **File**
- **Possesso File** importando l'email del proprietario dentro l'entità **File**
- **Possesso** importando l'identificare dell'utente proprietario dentro l'entità **Directory**
- **Versionamento** importando l'identificatore del file dentro l'entità **Versione** e utilizzandolo insieme all'attributo 'numero' come identificatore
- **Scaricamento Versione** importando l'identificativo della versione dentro l'entità **Download**
- **Visualizzazione Versione** importando l'identificativo della Versione dentro **Visualizzazione**
- **Scaricamento Utente** importando l'email dell'utente dentro l'entità **Download**
- **Visualizzazione Utente** importando l'email dell'utente dentro l'entità **Visualizzazione**
- **Apertura** importando l'email dell'utente dentro l'entità **Segnalazione**
- **Accettazione** importando il codice dell'operatore e la data di accettazione dentro l'entità **Segnalazione**
- **Chiusura** importando il codice dell'operatore e la data di chiusura dentro l'entità **Segnalazione**
- **Interessamento** importando l'identificatore della segnalazione dentro l'entità **Intervento**

- **Utente Interviene** importando l'email dell'utente dentro **Intervento** e rendendolo opzionale
- **Operatore Interviene** importando il codice dell'operatore dentro **Intervento** e rendendolo opzionale

3.5 Analisi delle ridondanze

3.5.1 Numero delle directory nell'utente

Si vuole decidere di salvarsi un attributo nell'utente che indica il numero di cartelle che ha attualmente in modo da velocizzare le operazioni 19 e 27.
L'aggiunta dell'attributo andrà però anche ad intaccare l'operazione 02.
Di seguito l'analisi delle due casistiche.

Senza Ridondanza

Operazione 02 - Creare una nuova directory

Concetto	Costrutto	Accesso	Tipo
Directory	E	1	S
Residenza	R	1	S
Possesso	R	1	S
Totale	(3S)* 5000 = 30000 al giorno		

Operazione 19 - Visualizzare il numero di cartelle per utente

Concetto	Costrutto	Accesso	Tipo
Utente	E	10000	L
Possesso	R	200000	L
Totale	(210000L)* 1 = 210000 al giorno		

Operazione 27 - Visualizzare per un utente il numero di directory

Concetto	Costrutto	Accesso	Tipo
Utente	E	1	L
Possesso	R	20	L
Totale	(21L)* 1000 = 21000 al giorno		

Senza la ridondanza abbiamo ottenuto un totale di **261.000** accessi al giorno.

Con Ridondanza

Operazione 02 - Creare una nuova directory

Concetto	Costrutto	Accesso	Tipo
Directory	E	1	S
Residenza	R	1	S
Possesso	R	1	S
Utente	E	1	L
Utente	E	1	S
Totale	(4S + 1L)* 5000 = 45000 al giorno		

Operazione 19 - Visualizzare il numero di cartelle per utente

Concetto	Costrutto	Accesso	Tipo
Utente	E	10000	L
Totale	(10000L)* 1 = 10000 al giorno		

Operazione 27 - Visualizzare per un utente il numero di directory

Concetto	Costrutto	Accesso	Tipo
Utente	E	1	L
Totale	(1L)* 1000 = 1000 al giorno		

Con la ridondanza abbiamo ottenuto un totale di **56.000** accessi al giorno.
Possiamo quindi procedere ad integrare la ridondanza poiché parecchio vantaggiosa.

3.5.2 Numero di versioni nel file

Si vuole decidere di salvare un attributo nel file che indica il numero di versioni che ha attualmente in modo da velocizzare l'operazione 25.
L'aggiunta dell'attributo andrà però anche ad intaccare l'operazione 05.
Di seguito l'analisi delle due casistiche.

Senza Ridondanza

Operazione 05 - Aggiungere una nuova versione di un file

Concetto	Costrutto	Accesso	Tipo
Versione	E	1	S
Versionamento	R	1	S
Totale	(2S)* 5000 = 20.000 al giorno		

Operazione 25 - Visualizzare per un file il numero di versioni

Concetto	Costrutto	Accesso	Tipo
File	E	1	L
Versionamento	R	3	L
Totale	(4L)* 10000 = 40.000 al giorno		

Senza la ridondanza abbiamo ottenuto un totale di **60.000** accessi al giorno.

Con Ridondanza

Operazione 05 - Aggiungere una nuova versione di un file

Concetto	Costrutto	Accesso	Tipo
Versione	E	1	S
Versionamento	R	1	S
File	E	1	L
File	E	1	S
Totale	(3S + 1L)* 5000 = 35.000 al giorno		

Operazione 25 - Visualizzare per un file il numero di versioni

Concetto	Costrutto	Accesso	Tipo
File	E	1	L
Totale	(1L)* 10000 = 10.000 al giorno		

Con la ridondanza abbiamo ottenuto un totale di **45.000** accessi al giorno.
In questo caso il guadagno sarebbe minimo, quindi si è deciso di evitare la ridondanza.

3.5.3 Numero delle sottocartelle in directory

Si vuole decidere se salvare un attributo nelle directory che indichi il numero di sottocartelle che ha attualmente in modo da velocizzare l'operazione 31.
L'aggiunta dell'attributo andrà però anche ad intaccare l'operazione 02.
Di seguito l'analisi delle due casistiche.

Senza Ridondanza

Operazione 02 - Creare una nuova directory

Concetto	Costrutto	Accesso	Tipo
Directory	E	1	S
Residenza	R	1	S
Possesso	R	1	S
Totale	(3S)* 5000 = 30000 al giorno		

Operazione 31 - Visualizzare il numero di cartelle in una cartella

Concetto	Costrutto	Accesso	Tipo
Directory	E	1	L
Residenza	R	3	L
Totale	(4L)* 2000 = 8000 al giorno		

Senza la ridondanza abbiamo ottenuto un totale di **38.000** accessi al giorno.

Con Ridondanza

Operazione 02 - Creare una nuova directory

Concetto	Costrutto	Accesso	Tipo
Directory	E	1	S
Residenza	R	1	S
Possesso	R	1	S
Directory	E	1	L
Directory	E	1	S
Totale	(4S + 1L)* 5000 = 45000 al giorno		

Operazione 31 - Visualizzare il numero di cartelle in una cartella

Concetto	Costrutto	Accesso	Tipo
Directory	E	1	L
Totale	(1L)* 2000 = 2000 al giorno		

Con la ridondanza abbiamo ottenuto un totale di **47.000** accessi al giorno.
In questo caso la ridondanza non farebbe altro che peggiorare la situazione, quindi non verrà applicata.

3.5.4 Associazione di possesso file da parte di un utente

Si vuole decidere se aggiungere una relazione ridondante in modo da collegare direttamente il file al proprietario senza dover passare dalla cartella in cui si trova. Questo gioverebbe molto all'operazione numero 26 ma soprattutto a tutte le ricerche di file di utenti in generale.
L'aggiunta della relazione andrà però anche ad intaccare l'operazione 03.
Di seguito l'analisi delle due casistiche.

Senza Ridondanza

Operazione 03 - Aggiungere un nuovo file

Concetto	Costrutto	Accesso	Tipo
File	E	1	S
Contenimento	R	1	S
Versione	E	1	S
Versionamento	R	1	S
Totale	(4S)* 10000 = 80000 al giorno		

Operazione 26 - Visualizzare per un utente il numero di file

Concetto	Costrutto	Accesso	Tipo
Utente	E	1	L
Possesso	R	20	L
Directory	E	20	L
Contenimento	R	100	L
Totale	(141L)* 1000 = 141000 al giorno		

Senza la ridondanza abbiamo ottenuto un totale di **221.000** accessi al giorno.

Con Ridondanza

Operazione 03 - Aggiungere un nuovo file

Concetto	Costrutto	Accesso	Tipo
File	E	1	S
Contenimento	R	1	S
Versione	E	1	S
Versionamento	R	1	S
Possesso File	R	1	S
Totale	(5S)* 10000 = 90000 al giorno		

Operazione 26 - Visualizzare per un utente il numero di file

Concetto	Costrutto	Accesso	Tipo
Utente	E	1	L
Possesso File	R	100	L
Totale	(101L)* 1000 = 101000 al giorno		

Con la ridondanza abbiamo ottenuto un totale di **191.000** accessi al giorno.
Come si può vedere dai risultati abbiamo un guadagno nell'utilizzare la ridondanza e seppur non sia un grande guadagno si vuole decidere di mantenere la ridondanza poiché questa gioverà qualsiasi operazione futura il cui meccanismo è la ricerca di file di uno specifico utente.

3.5.5 Relazione di ultima versione di un file

Si vuole decidere se aggiungere una relazione ridondante in modo da collegare direttamente il file alla sua versione, la quale determina il peso attuale del file oltre ad avere al suo interno altri dati importanti. Questo gioverebbe in particolare alle operazioni 33 e 34.
L'aggiunta della relazione andrà però anche ad intaccare l'operazione 03 e l'operazione 05. Di seguito l'analisi delle due casistiche.

Senza Ridondanza

Operazione 03 - Aggiungere un nuovo file

Concetto	Costrutto	Accesso	Tipo
File	E	1	S
Contenimento	R	1	S
Versione	E	1	S
Versionamento	R	1	S
Totale	(4S)* 10000 = 80000 al giorno		

Operazione 05 - Aggiungere una nuova versione di un file

Concetto	Costrutto	Accesso	Tipo
Versione	E	1	S
Versionamento	R	1	S
Totale	(2S)* 5000 = 20000 al giorno		

Operazione 33 - Visualizzare la dimensione di un file

Concetto	Costrutto	Accesso	Tipo
File	E	1	L
Versionamento	R	3	L
Versione	E	3	L
Totale	(7L)* 20000 = 140000 al giorno		

Operazione 34 - Visualizzare la dimensione di una cartella

Concetto	Costrutto	Accesso	Tipo
Directory	E	1	L
Contenimento	R	5	L
File	E	5	L
Versionamento	R	15	L
Versione	E	15	L
Totale	(41L)* 5000 = 205000 al giorno		

Senza la ridondanza abbiamo ottenuto un totale di **445.000** accessi al giorno.

Con Ridondanza

Operazione 03 - Aggiungere un nuovo file

Concetto	Costrutto	Accesso	Tipo
File	E	1	S
Contenimento	R	1	S
Versione	E	1	S
Versionamento	R	1	S
Ultima Versione	R	1	S
Totale	(5S)* 10000 = 100000 al giorno		

Operazione 05 - Aggiungere una nuova versione di un file

Concetto	Costrutto	Accesso	Tipo
Versione	E	1	S
Versionamento	R	1	S
Ultima Versione	R	1	S
Totale	(3S)* 5000 = 30000 al giorno		

Operazione 33 - Visualizzare la dimensione di un file

Concetto	Costrutto	Accesso	Tipo
File	E	1	L
Ultima Versione	R	1	L
Versione	E	1	L
Totale	(3L)* 20000 = 60000 al giorno		

Operazione 34 - Visualizzare la dimensione di una cartella

Concetto	Costrutto	Accesso	Tipo
Directory	E	1	L
Contenimento	R	5	L
File	E	5	L
Ultima Versione	R	5	L
Versione	E	5	L
Totale	(21L)* 5000 = 105000 al giorno		

Con la ridondanza abbiamo ottenuto un totale di **295.000** accessi al giorno.
Come si può vedere dai risultati abbiamo un guadagno piuttosto buono, anche contando che la relazione ridondante risulta molto utile per un innumerevole numero di operazioni che coinvolgono solamente l'ultima versione di un file e che non si occupano delle versioni precedenti.

3.6 Traduzione di entità e associazioni in relazioni

Directories(Id, Nome, DataCreazione, Padre*, Proprietario)
FK: Padre REFERENCES Directories
FK: Proprietario REFERENCES Utenti
Unique(Padre, Nome)

Files(Id, Directory, Nome, Estensione, Proprietario, UltimaVersione)
FK: Directory REFERENCES Directories
FK: Proprietario REFERENCES Utenti
FK: UltimaVersione REFERENCES Versioni
UNIQUE(Directory, Nome, Estensione)

Utenti(Email, Nome, Cognome, DataRegistrazione, Password, DataNascita, NumeroDirectory)

Preferenze(File, Utente, DataPreferenza)
FK: File REFERENCES Files
FK: Utente REFERENCES Utenti

Versioni(Id, File, Numero, DataCreazione, Dimensione, Link)
FK: File REFERENCES Files
UNIQUE(File, Numero)

Downloads(Id, Versione, Utente, DataDownload)
FK: Versione REFERENCES Versioni
FK: Utente REFERENCES Utenti

Visualizzazioni(Id, Versione, Utente, DataVisualizzazione)
FK: Versione REFERENCES Versioni
FK: Utente REFERENCES Utenti

Segnalazioni(Id, Utente, Descrizione, Operatore*, DataAccettazione*, DataChiusura*)
FK: Utente REFERENCES Utenti
FK: Operatore REFERENCES Operatori

Operatori(Codice, Nome, Password, DataNascita)

Interventi(Segnalazione, Numero, Utente*, Operatore*, Messaggio, DataIntervento)
FK: Segnalazione REFERENCES Segnalazioni
FK: Utente REFERENCES Utenti
FK: Operatore REFERENCES Operatori
CHECK (

(Operatore is null or Utente is null)
and not
(Operatore is null and Utente is null)
)

Condivisioni(File, Utente, Scrittura, DataCondivisione)
FK: File REFERENCES Files
FK: Utente REFERENCES Utenti

3.7 Schema relazionale finale

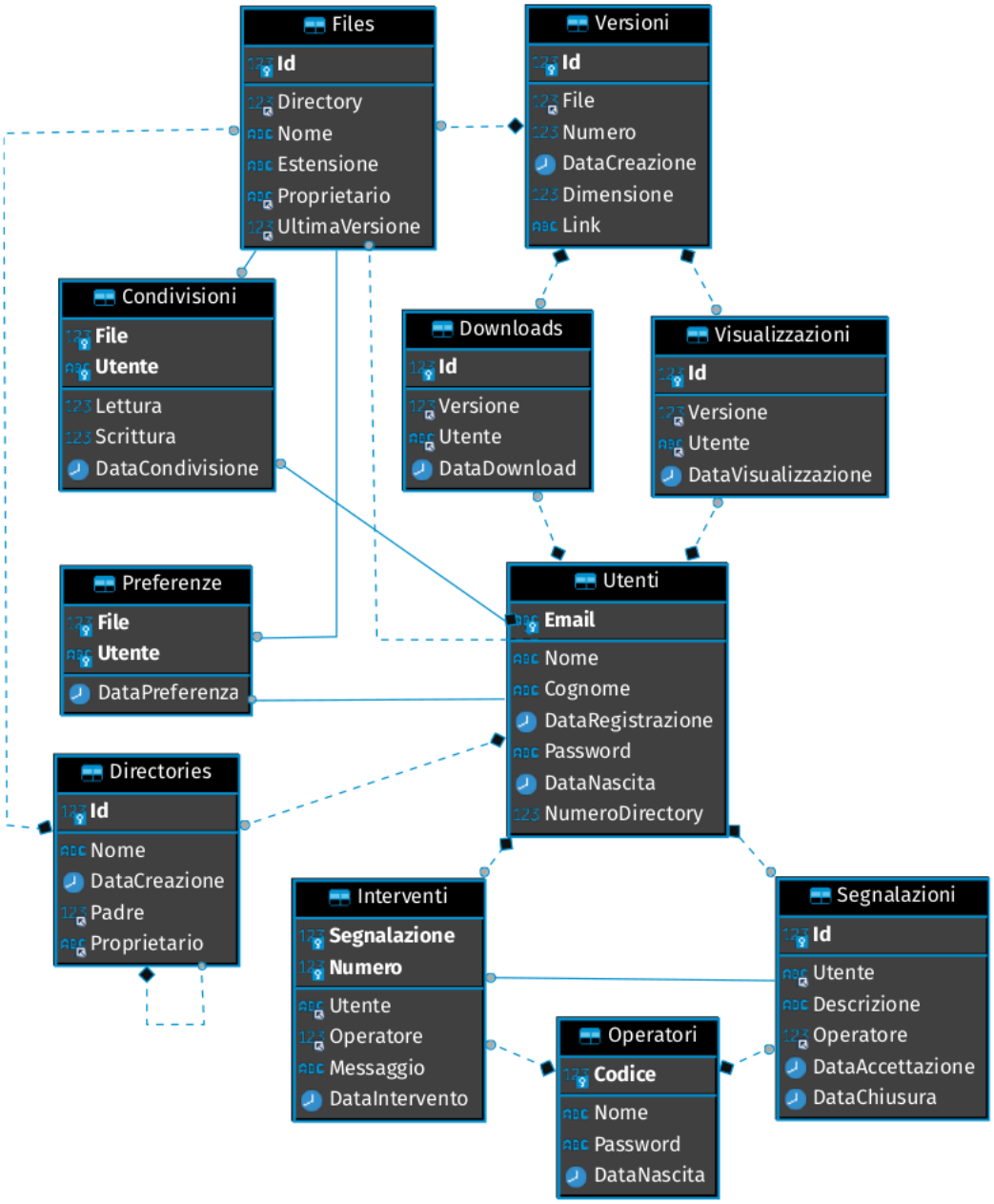

Figura 3.2: Schema Relazionale

3.8 Costruzione delle tabelle in linguaggio SQL

```
# Create schemas

# Create tables
CREATE TABLE IF NOT EXISTS Directories
(
    Id INT NOT NULL AUTO_INCREMENT,
    Nome VARCHAR(80) NOT NULL,
    DataCreazione DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    Padre INT,
    Proprietario VARCHAR(50) NOT NULL,
    PRIMARY KEY(Id),
    UNIQUE(Padre, Nome)
);

CREATE TABLE IF NOT EXISTS Files
(
    Id INT NOT NULL AUTO_INCREMENT,
    Directory INT NOT NULL,
    Nome VARCHAR(80) NOT NULL,
    Estensione VARCHAR(10) NOT NULL,
    Proprietario VARCHAR(50),
    UltimaVersione INT,
    PRIMARY KEY(id),
    UNIQUE(Directory, Nome, Estensione)
);

CREATE TABLE IF NOT EXISTS Utenti
(
    Email VARCHAR(50) NOT NULL,
    Nome VARCHAR(20) NOT NULL,
    Cognome VARCHAR(20) NOT NULL,
    DataRegistrazione DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    Password VARCHAR(40) NOT NULL,
    DataNascita DATE NOT NULL,
    NumeroDirectory INT NOT NULL DEFAULT 0,
    PRIMARY KEY(Email)
);


CREATE TABLE IF NOT EXISTS Preferenze
(
    File INT NOT NULL,
    Utente VARCHAR(50) NOT NULL,
    DataPreferenza DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY(File, Utente)
);

CREATE TABLE IF NOT EXISTS Versioni
(
```

```
50      Id INT NOT NULL AUTO_INCREMENT,
51      File INT NOT NULL,
52      Numero INT,
53      DataCreazione DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
54      Dimensione INT NOT NULL,
55      Link VARCHAR(100) NOT NULL,
56      PRIMARY KEY(Id),
57      UNIQUE(File, Numero),
58
59  );
60
61  CREATE TABLE IF NOT EXISTS Downloads
62  (
63    Id INT NOT NULL AUTO_INCREMENT,
64      Versione INT NOT NULL,
65      Utente VARCHAR(50) NOT NULL,
66      DataDownload DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
67      PRIMARY KEY(Id)
68  );
69
70  CREATE TABLE IF NOT EXISTS Visualizzazioni
71  (
72    Id INT NOT NULL AUTO_INCREMENT,
73      Versione INT NOT NULL,
74      Utente VARCHAR(50) NOT NULL,
75      DataVisualizzazione DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
76      PRIMARY KEY(Id)
77  );
78
79  CREATE TABLE IF NOT EXISTS Segnalazioni
80  (
81      Id INT NOT NULL AUTO_INCREMENT,
82      Utente VARCHAR(50) NOT NULL,
83      Descrizione VARCHAR(1000) NOT NULL,
84      Operatore INT,
85      DataAccettazione DATETIME,
86      DataChiusura DATETIME,
87      PRIMARY KEY(Id)
88  );
89
90  CREATE TABLE IF NOT EXISTS Operatori
91  (
92      Codice INT NOT NULL,
93      Nome VARCHAR(20) NOT NULL,
94      Password VARCHAR(20) NOT NULL,
95      DataNascita DATE NOT NULL,
96      PRIMARY KEY(Codice)
97  );
98
99  CREATE TABLE IF NOT EXISTS Interventi
100 (
```

```
    Segnalazione INT NOT NULL,
    Numero INT NOT NULL,
    Utente VARCHAR(50),
    Operatore INT,
    Messaggio VARCHAR(1000) NOT NULL,
    DataIntervento DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY(Segnalazione, Numero),
    CONSTRAINT GERARCHIA_INTERVENTI CHECK (
      (Operatore is null or Utente is null)
      and not
      (Operatore is null and Utente is null)
    )

);

CREATE TABLE IF NOT EXISTS Condivisioni
(
    File INT NOT NULL,
    Utente VARCHAR(50) NOT NULL,
    Lettura BOOL NOT NULL,
    Scrittura BOOL NOT NULL,
    DataCondivisione DATE DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY(File, Utente),
    CONSTRAINT PERMESSI_VALIDI CHECK (
      (Lettura = TRUE)
    )
);


# Create FKs
ALTER TABLE Files
    ADD    FOREIGN KEY (Directory)
    REFERENCES Directories(Id)
;

ALTER TABLE Directories
    ADD    FOREIGN KEY (Padre)
    REFERENCES Directories(Id)
;

ALTER TABLE Directories
    ADD    FOREIGN KEY (Proprietario)
    REFERENCES Utenti(Email)
;

ALTER TABLE Preferenze
    ADD    FOREIGN KEY (Utente)
    REFERENCES Utenti(Email)
;

ALTER TABLE Downloads
```

```
    ADD    FOREIGN KEY (Versione)
    REFERENCES Versioni(Id)
;

ALTER TABLE Downloads
    ADD    FOREIGN KEY (Utente)
    REFERENCES Utenti(Email)
;

ALTER TABLE Visualizzazioni
    ADD    FOREIGN KEY (Utente)
    REFERENCES Utenti(Email)
;

ALTER TABLE Segnalazioni
    ADD    FOREIGN KEY (Utente)
    REFERENCES Utenti(Email)
;

ALTER TABLE Segnalazioni
    ADD    FOREIGN KEY (Operatore)
    REFERENCES Operatori(Codice)
;

ALTER TABLE Interventi
    ADD    FOREIGN KEY (Segnalazione)
    REFERENCES Segnalazioni(Id)
;

ALTER TABLE Interventi
    ADD    FOREIGN KEY (Utente)
    REFERENCES Utenti(Email)
;

ALTER TABLE Interventi
    ADD    FOREIGN KEY (Operatore)
    REFERENCES Operatori(Codice)
;

ALTER TABLE Condivisioni
    ADD    FOREIGN KEY (Utente)
    REFERENCES Utenti(Email)
;

ALTER TABLE Files
    ADD    FOREIGN KEY (Proprietario)
    REFERENCES Utenti(Email)
;

ALTER TABLE Files
    ADD    FOREIGN KEY (UltimaVersione)
```

```
203     REFERENCES Versioni(Id)
204 ;
205
206 ALTER TABLE Preferenze
207     ADD    FOREIGN KEY (File)
208     REFERENCES Files(Id)
209 ;
210
211 ALTER TABLE Condivisioni
212     ADD    FOREIGN KEY (File)
213     REFERENCES Files(Id)
214 ;
215
216 ALTER TABLE Visualizzazioni
217     ADD    FOREIGN KEY (Versione)
218     REFERENCES Versioni(Id)
219 ;
220
221 # Create Triggers
222
223 CREATE DEFINER=`root`@`localhost` TRIGGER UTENTE_MAGGIORENNE
224 BEFORE INSERT
225 ON Utenti FOR EACH ROW
226 BEGIN
227 IF YEAR(curdate()) - YEAR(NEW.DataNascita) - (DATE_FORMAT(curdate(), "%m%
        d") < DATE_FORMAT(NEW.DataNascita, "%m%d")) < 18
228 THEN SIGNAL SQLSTATE "45000" SET MESSAGE_TEXT = "Gli utenti devono essere
        maggiorenni";
229 END IF;
230 END;
231
232 CREATE DEFINER=`root`@`localhost` TRIGGER PROPRIETARIO_FILE_AUTOMATICO
233 BEFORE INSERT
234 ON Files FOR EACH ROW
235 BEGIN
236   SET NEW.Proprietario = (SELECT Proprietario FROM Directories WHERE Id =
        NEW.Directory);
237 END
238
239 CREATE DEFINER=`root`@`localhost` TRIGGER
        PROPRIETARIO_DIRECTORY_AUTOMATICO
240 BEFORE INSERT
241 ON Directories FOR EACH ROW
242 BEGIN
243   DECLARE Prop VARCHAR(50);
244   SET Prop = (SELECT Proprietario FROM Directories WHERE Id = NEW.Padre);
245   IF not (Prop is NULL)
246   THEN
247     SET NEW.Proprietario = Prop;
248   END IF;
249 END
```

```
250
251 CREATE DEFINER=`root`@`localhost` TRIGGER INCREMENTO_VERSIONE
252 BEFORE INSERT
253 ON Versioni FOR EACH ROW
254 BEGIN
255   DECLARE Prev INT;
256 IF NEW.Numero is NULL
257 THEN
258   SET Prev = (SELECT MAX(Numero) FROM Versioni WHERE File = NEW.File);
259   IF Prev is NULL
260   THEN
261     SET NEW.Numero = 1;
262   ELSE
263     SET NEW.Numero = Prev + 1;
264   END IF;
265 END IF;
266 END;
267
268 CREATE DEFINER=`root`@`localhost` TRIGGER AGGIORNAMENTO_ULTIMA_VERSIONE
269 AFTER INSERT
270 ON Versioni FOR EACH ROW
271 BEGIN
272 UPDATE Files SET UltimaVersione = NEW.Id WHERE Id = NEW.File;
273 END;
274
275 CREATE DEFINER=`root`@`localhost` TRIGGER UPDATE_UTENTE
276 AFTER INSERT
277 ON Directories FOR EACH ROW
278 BEGIN
279   UPDATE Utenti SET NumeroDirectory = NumeroDirectory + 1 WHERE Email =
      NEW.Proprietario;
280 END
281
282 CREATE DEFINER=`root`@`localhost` TRIGGER INCREMENTO_INTERVENTO
283 BEFORE INSERT
284 ON Interventi FOR EACH ROW
285 BEGIN
286   DECLARE Prev INT;
287 IF NEW.Numero is NULL
288 THEN
289   SET Prev = (SELECT MAX(Numero) FROM Interventi WHERE Segnalazione = NEW
      .Segnalazione);
290   IF Prev is NULL
291   THEN
292     SET NEW.Numero = 1;
293   ELSE
294     SET NEW.Numero = Prev + 1;
295   END IF;
296 END IF;
297 END
298
```

```
## Possibili constraint che sono stati evitati per non complicare la gestione dell applicazione
/*

TABLE: Preferenze
CONSTRAINT PREFERENZA_VALIDA CHECK (
  ((SELECT Proprietario FROM Files WHERE Id = File) = Utente)
  or
  EXISTS (SELECT * FROM Condivisioni c WHERE c.File = File AND c.Utente = Utente)
)


TABLE: Downloads
CONSTRAINT DOWNLOAD_VALIDO CHECK (
  ((SELECT Proprietario FROM Files WHERE Id = File) = Utente)
  or
  EXISTS (SELECT * FROM Condivisioni c WHERE c.File = File AND c.Utente = Utente)
)


TABLE: Visualizzazioni
CONSTRAINT VISUALIZZAZIONE_VALIDA CHECK (
  ((SELECT Proprietario FROM Files WHERE Id = File) = Utente)
  or
  EXISTS (SELECT * FROM Condivisioni c WHERE c.File = File AND c.Utente = Utente)
)
*/
```

Listing 3.1: DDL

3.9 Traduzione delle operazioni in query SQL

```
/* Operazione 01 - Registrare un nuovo utente */
INSERT INTO Utenti (Email, Nome, Cognome, Password, DataNascita) VALUES (?, ?, ?, ?, ?)
INSERT INTO Directories (Nome, Proprietario) VALUES (?, ?)
/* quando viene creato l utente viene creata anche la sua cartella principale */


/* Operazione 02 - Creare una nuova directory */
INSERT INTO Directories (Nome, Padre) VALUES (?, ?)
/*
il trigger prende il proprietario dalla cartella padre che non dovr essere nulla
il trigger incrementa il numero delle directory del proprietario da solo
*/


/* Operazione 03 - Aggiungere un nuovo file */
SET IdFile = INSERT INTO Files (Directory, Nome, Estensione) VALUES (?, ?, ?)
INSERT INTO Versioni (File, Dimensione, Link) VALUES (IdFile, ?, ?)
/*
il trigger setta automaticamente il proprietario del file prendendolo dalla directory
il trigger setta automaticamente la versione giusta incrementando il precedente
il trigger va ad aggiorna il campo UltimaVersione di File
*/

/* Operazione 04 - Rinominare un file */
UPDATE Files SET Nome = ? and Estensione = ? WHERE Id = ?


/* Operazione 05 - Aggiungere una nuova versione di un file */
INSERT INTO Versioni (File, Dimensione, Link) VALUES (?, ?, ?)
/*
il trigger setta automaticamente la versione giusta incrementando il precedente
L altro trigger va ad aggiornare la relazione all ultima versione nel file
*/

/* Operazione 06 - Scaricare una versione di un file */
INSERT INTO Downloads (Versione, Utente) VALUES (?, ?)


/* Operazione 07 - Aggiungere un file ai preferiti */
INSERT INTO Preferenze (File, Utente) VALUES (?, ?)

```

```
42
43 /* Operazione 08 - Condividere un file con un utente */
44 INSERT INTO Condivisioni (File, Utente, Lettura, Scrittura) VALUES (?, ?,
       ?, ?)
45
46
47 /* Operazione 09 - Visualizzare un file */
48 INSERT INTO Visualizzazioni (Versione, Utente) VALUES (?, ?)
49
50
51 /* Operazione 10 - Aprire una segnalazione */
52 INSERT INTO Segnalazioni (Utente, Descrizione) VALUES (?, ?)
53
54
55 /* Operazione 11 - Accettare una segnalazione */
56 UPDATE Segnalazioni SET Operatore = ?, DataAccettazione = NOW() WHERE Id
       = ?
57
58
59 /* Operazione 12 - Chiudere una segnalazione */
60 UPDATE Segnalazioni SET DataChiusura = NOW() WHERE Id = ?
61
62
63 /* Operazione 13 - Fare un intervento da parte di un utente */
64 INSERT INTO Interventi (Segnalazione, Utente, Messaggio) VALUES (?, ?, ?)
65 /* il trigger gestisce automatica la sequenza numerica */
66
67
68 /* Operazione 14 - Fare un intervento da parte di un operatore */
69 INSERT INTO Interventi (Segnalazione, Operatore, Messaggio) VALUES (?, ?,
       ?)
70 /* il trigger gestisce automatica la sequenza numerica */
71
72
73 /* Operazione 15 - Registrare un nuovo operatore */
74 INSERT INTO Operatori (Codice, Nome, DataNascita, Password) VALUES (?, ?,
       ?, ?)
75
76
77 /* Operazione 16 - Visualizzare  l operatore  che ha chiuso pi
       interventi */
78 SELECT Operatore
79 FROM Segnalazioni
80 WHERE not (DataChiusura is null)
81 GROUP BY Operatore
82 ORDER BY COUNT(*) DESC
83 LIMIT 1
84
85
86 /* Operazione 17 - Visualizzare il tipo di file pi  presente */
87 SELECT Estensione
```

```sql
FROM Files
GROUP BY Estensione
ORDER BY COUNT(*) DESC
LIMIT 1

/* Operazione 18 - Visualizzare il numero di file preferiti per utente */
SELECT u.*, COUNT(*) as NumeroPreferiti
FROM Utenti u inner join Preferenze p on u.Email = p.Utente
GROUP BY u.Email


/* Operazione 19 - Visualizzare il numero di cartelle per utente */
SELECT u.Email, u.NumeroDirectory
FROM Utenti u


/* Operazione 20 - Visualizzare il file pi  condiviso */
SELECT f.*
FROM Files f inner join Condivisioni c on f.Id = c.File
GROUP BY f.Id
ORDER BY COUNT(*) DESC
LIMIT 1


/* Operazione 21 - Visualizzare il file pi  visualizzato */
SELECT f.*
FROM Files f, Versioni v, Visualizzazioni vi
WHERE f.Id = v.File and v.Id = vi.Versione
GROUP BY f.Id
ORDER BY COUNT(*) DESC
LIMIT 1


/* Operazione 22 - Visualizzare  l utente  che ha aperto pi
   segnalazioni */
SELECT *
FROM Utenti u inner join Segnalazioni s on u.Email = s.Utente
GROUP BY f.Id
ORDER BY COUNT(*) DESC
LIMIT 1


/* Operazione 23 - Visualizzare il file pi  scaricato */
SELECT f.*
FROM Files f, Versioni v, Downloads d
WHERE f.Id = v.File and v.Id = d.Versione
GROUP BY f.Id
ORDER BY COUNT(*) DESC
LIMIT 1


```

```sql
/* Operazione 24 - Visualizzare tutti i file scaricati di un utente */
SELECT DISTINCT f.*
FROM Utenti u, Downloads d, Versioni v, Files f
WHERE u.Email = d.Utente and d.Versione = v.Id and v.File = f.Id
and f.Proprietario = ?


/* Operazione 25 - Visualizzare per un file il numero di versioni */
SELECT COUNT(*) as NumeroVersioni
FROM Files f inner join Versioni v on f.Id = v.File
WHERE f.Id = ?


/* Operazione 26 - Visualizzare per un utente il numero di file */
SELECT COUNT(*) as NumeroFile
FROM Utenti u inner join File f on u.Email = f.Proprietario
WHERE u.Email = ?


/* Operazione 27 - Visualizzare per un utente il numero di directory */
SELECT NumeroDirectory
FROM Utenti
WHERE u.Email = ?


/* Operazione 28 - Visualizzare per un utente il numero di file preferiti
    */
SELECT COUNT(*) as NumeroPreferenze
FROM Utenti u inner join Preferenze p on u.Email = p.Utente
WHERE u.Email = ?


/* Operazione 29 - Visualizzare il numero di file in una cartella */
SELECT COUNT(*) as NumeroFile
FROM Directories d inner join Files f on d.Id = f.Directory
WHERE d.Id = ?


/* Operazione 30 - Visualizzare i file in una cartella */
SELECT f.*
from Directories d inner join Files f on d.Id = f.Directory
WHERE d.Id = ?


/* Operazione 31 - Visualizzare il numero di cartelle in una cartella */
SELECT COUNT(*) as NumeroSottoCartelle
FROM Directories d inner join Directories d2 on d.Id = d2.Padre
WHERE d.Id = ?


/* Operazione 32 - Visualizzare le cartelle in una directory */
```

```
188 SELECT d.*
189 FROM Directories d
190 WHERE d.Padre = ?
191
192
193 /* Operazione 33 - Visualizzare la dimensione di un file */
194 SELECT f.*, v.Dimensione
195 FROM Files f inner join Versioni v on v.Id = f.UltimaVersione
196 WHERE f.Id = ?
197
198
199 /* Operazione 34 - Visualizzare la dimensione di una cartella */
200 SELECT sum(v.Dimensione) as Dimensione
201 FROM Directories d, Files f, Versioni v
202 WHERE d.Id = f.Directory and f.UltimaVersione = v.Id and d.Id = ?
```

Listing 3.2: Query

Capitolo 4

Progettazione dell'applicazione

Per lo sviluppo dell'applicazione si è scelto di utilizzare come database MySQL e come linguaggio Java insieme al framework JavaFX in modo da creare in modo semplice un software che risultasse cross-platform.
L'architettura dell'applicazione usa come basa quello del progetto creato nel corso di programmazione ad oggetti visualizzabile al seguente link https://github.com/zucchero-sintattico/OOP20-Jhaturanga.
Di seguito ne verranno approfondite le funzionalità e l'architettura generale.

4.1 Architettura

4.1.1 MVC

Come pattern architetturale è stato utilizzato MVC (Model - View - Controller).
In questo caso il model è composta da tutte le entità e delle metodologie di accesso ad esse e si occupa inoltre di tenere in memoria gli oggetti che servono da connessione con le entità del database in modo da poterle selezionare ed aggiungere in modo molto semplice.
Le View sono le pagine visualizzabili e che con le quali l'utente interagisce, sarà quindi loro responsabilità di effettuare le necessarie operazioni in risposta all'input dell'operatore.
I Controller invece sono gli agenti che si occupano di effettuare le operazioni sul Model e quindi sui dati per poi restituirli alla View che si occupa di mostrarli.
In questo caso abbiamo che ad ogni View corrisponde un proprio Controller che gestirà solamente la parte di Model inerente a cioò che la View sta mostrando.
Per una spiegazione più dettagliata della struttura MVC si consiglia di leggere la parte di architettura del progetto Jhaturanga.

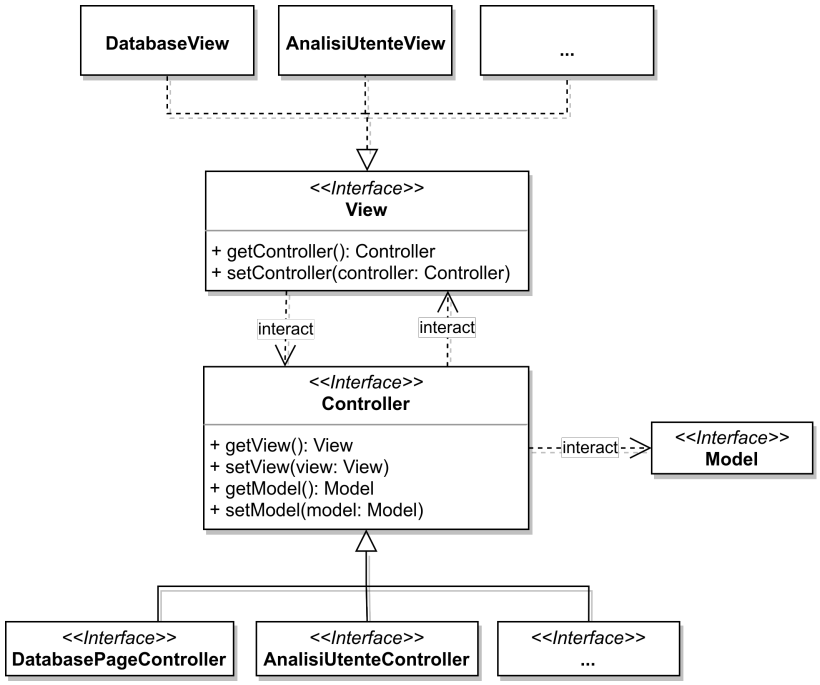

Figura 4.1: Architettura dell'applicazione

4.1.2 Query

Nello sviluppo dell'applicativo si è cercato di sviluppare un ORM personale, è quindi stato necessario fare un breakdown delle entità coinvolte in gioco.
Una delle prime entità che hanno preso forma è stata quella della query, la quale tramite l'oggetto **Query** mantiene la struttura dei paramentri internamente e nel momento della chiamata al metodo *toSql* restituisce la query sotto forma di stringa SQL.
Per la costruzione dell'oggetto invece si è scelto di utilizzare il pattern **Builder** in modo **fluente**.
Ciò ha permesso che la creazione delle query fosse fatta in modo molto snello e pulito, tuttavia è importante ricordare che quello che ne è risultato è solamente una semplificazione di un creatore di query poiché nonostante le sue funzionalità siano bastate per lo sviluppo dell'applicativo risulta comunque molto limitato.

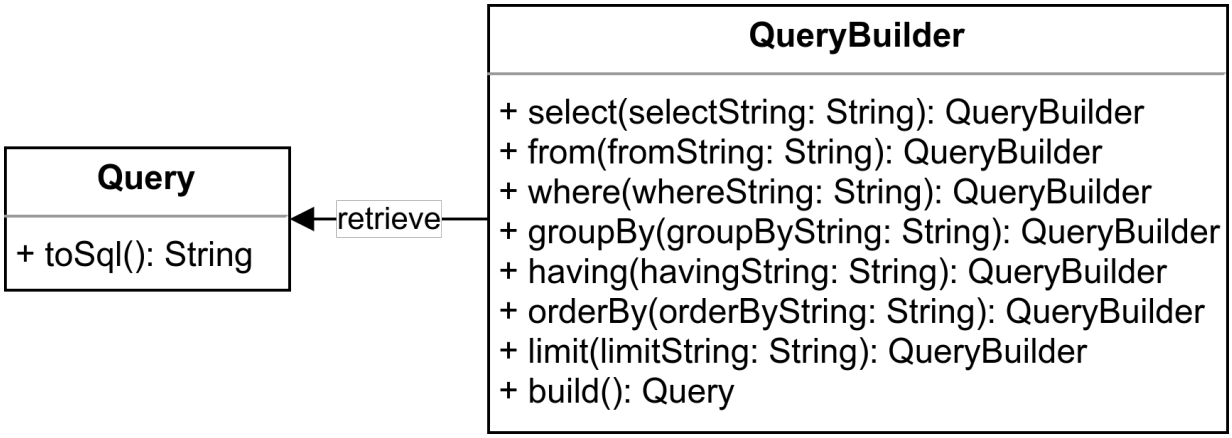

Figura 4.2: Gestione Query

4.1.3 Entità

Per la gestione delle entità **QueryObjectResult** che rappresenta un qualsiasi oggetti risultato di una query del database.
L'unica cosa che deve avere per ora è il metodo *toString* che tramite l'uso della Reflection stampa tutti i campi pubblici dell'oggetto che estende quella classe.
Da questa oggeto base deriva poi la classe **Entity** il cui scopo è quello di essere un entità contenuta nel database, infatti ogni entità dovrà implementare il metodo che ritorna il nome della Table in cui si trova e dovrà fornire su richiesta la versione base della sua query di INSERT.
Parte importante dell'architettura è l'oggetto **ObjectMapper** che consiste nel cuore dell ORM poiché permette (in modo per ora semplificato) di mappare un risultato di una query in un oggetto tramite l'uso della Reflection.

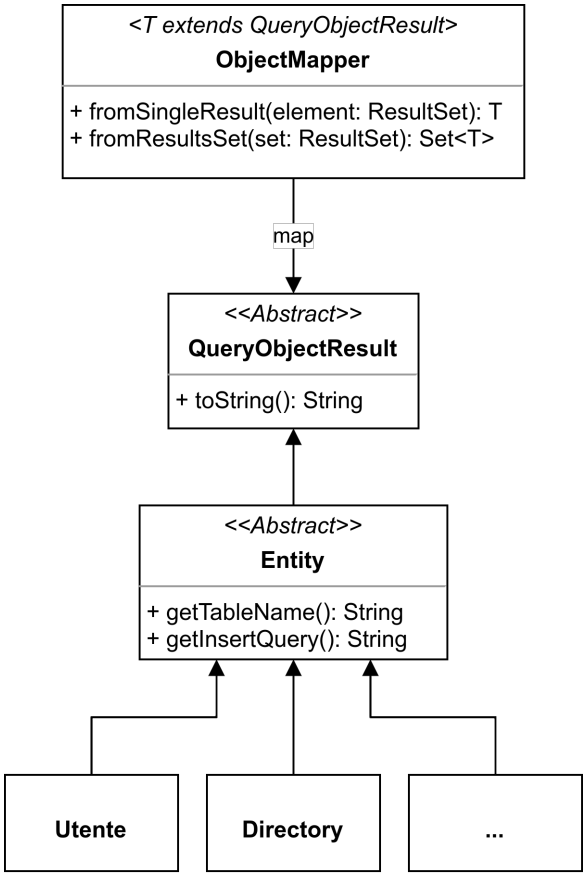

Figura 4.3: Gestione entità

4.1.4 Connessione Entità

Per l'interfacciamento con l'insieme delle istanze degli oggetti nel database è stato utilizzato la classe **DatabaseEntityConnection** che tramite la specifica dell'entità a cui fa rifermento e tramite l'utilizzo di ObjectMapper permette di ottenere tutti i risultati di una query su quell'entità e permette anche di inserire una nuova istanza nel database.
L'inserimento, cosi come tutti i vari aggiornamenti/cancellazioni sono effettuati tramite una **DatabaseOperation** che consiste in un wrapper per effettuare operazioni SQL.
Un esempio di funzionalità che è tornata molto comoda è stata l'implementazione del metodo *executeAndGetGeneratedKey* che se usato per eseguire una query di insert ti restituisce la chiave dell'oggetto generato.
La sua utilità si è rivelata nell'inserimento di entità che utilizzano come chiave un identificatore auto incrementale.

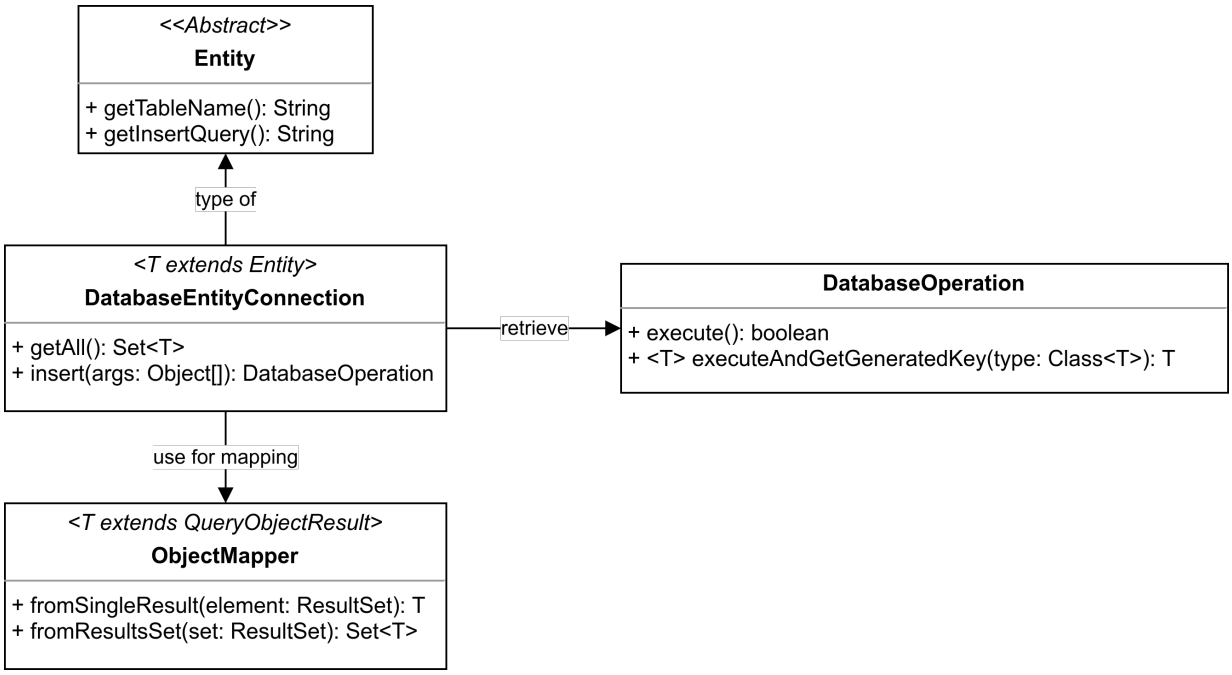

Figura 4.4: Gestione interfacciamento entità

4.1.5 Operazioni

Dopo avere già esplicitato quali sono le operazioni che la piattaforma deve potere effettuare possiamo notare che quelle di normale gestione di inserimento/lettura/modifica delle singole entità sono implementabili tramite cio che è stato spiegato precedentemente.
Per quanto riguarda invece le operazioni di analisi che coinvolgono quindi query un po più particolari si è partiti dall'oggetto **DatabaseQuery** che semplifica tramite l'utilizzo dei generici l'esecuzione di una query e il mapping del risultato.

Infatti vorrà come parametro il tipo dell'oggetto risultato che sarà poi utilizzato internamente da un **ObjectMapper** per il mapping.
L'insieme delle operazioni è stato quindi diviso con due Enum, una contenente le operazioni di analisi di CloudStore che coinvolgono tutto il database e una che contiene le operazioni di analisi che può svolgere un utente normale, limitata quindi a ciò che è di sua proprietà.
Per poter utilizzare l'entità **DatabaseQuery** è stato quindi necessario realizzare alcuni oggetti denominati **QueryXXResult** dove XX è il numero della query a cui corrispondono il cui compito è quello di rappresentare il risultato della query in modo da avere un oggetto che tramite l'**ObjectMapper** potesse essere generato dal risultato dell'operazione.

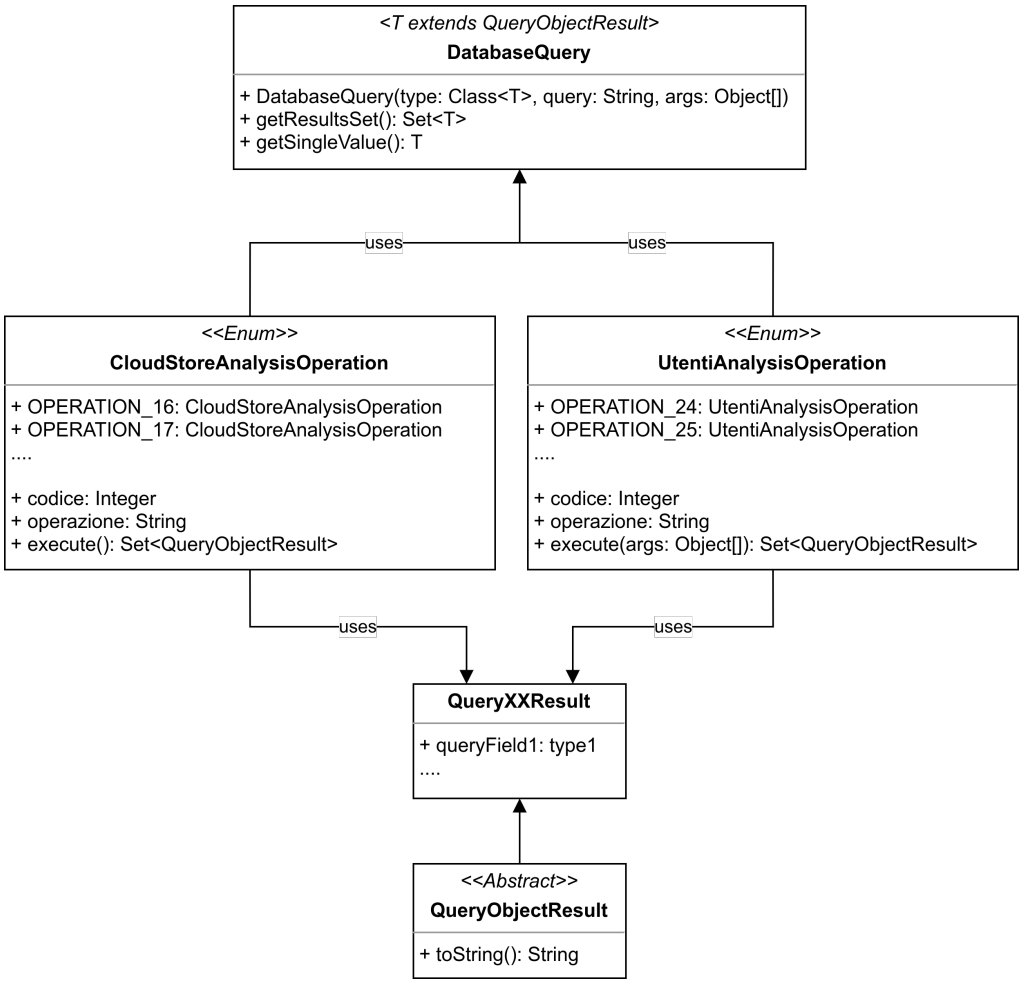

Figura 4.5: Gestione operazioni generiche del database

4.2 Guida Utente

L'applicazione si divide in 3 sezioni principali che andremo ad elencare a spiegare di seguito.

4.2.1 Home Page

All'avvio dell'applicazione ci troveremo nella homepage che ci permetterà di selezionare la sezione dell'app in cui vogliamo andare.
Ci basterà cliccare sul bottone apposito e verremo reindirizzati alla sezione dedicata.
Per la navigazione tra le pagine sarà poi reso disponibile in alto a sinistra un menu a tendina che permetterà di selezionare la pagina a cui si vuole andare.

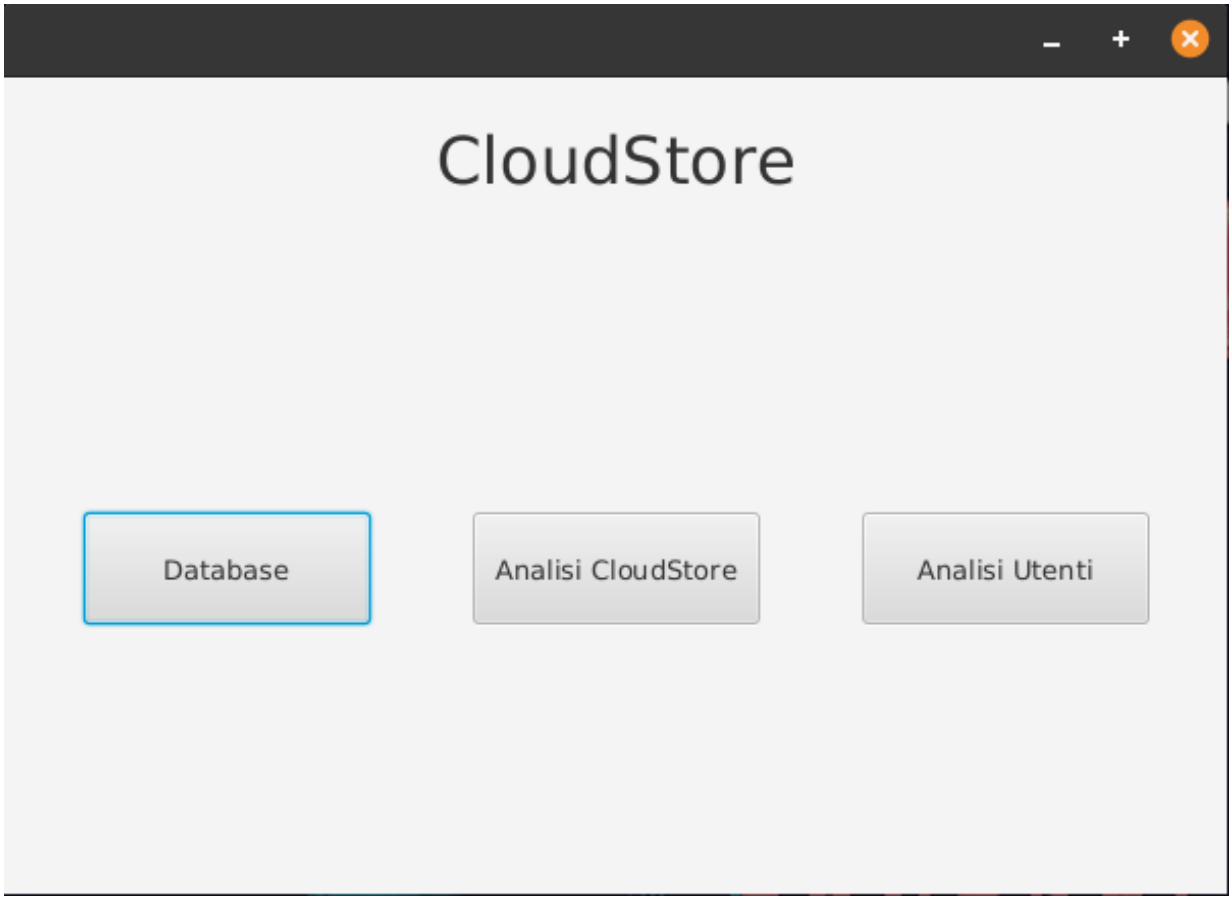

Figura 4.6: Homepage dell'applicazione

4.2.2 Database

La sezione denominata *Database* è quella dove possiamo vedere lo stato attuale del database e delle entità da cui è composto.
In questa pagina troviamo una Tab per ciascuna di queste entità.
In ognuna delle Tab abbiamo una lista dedicata a mostrare l'insieme dei record relativi a quella tabella mentre sulla destra troviamo la sezione dedicata all'inserimento di una nuova istanza di quell'entità.
Per poter procedere all'inserimento sarà quindi necessario inserire dei valori in tutti i campi richiesti.

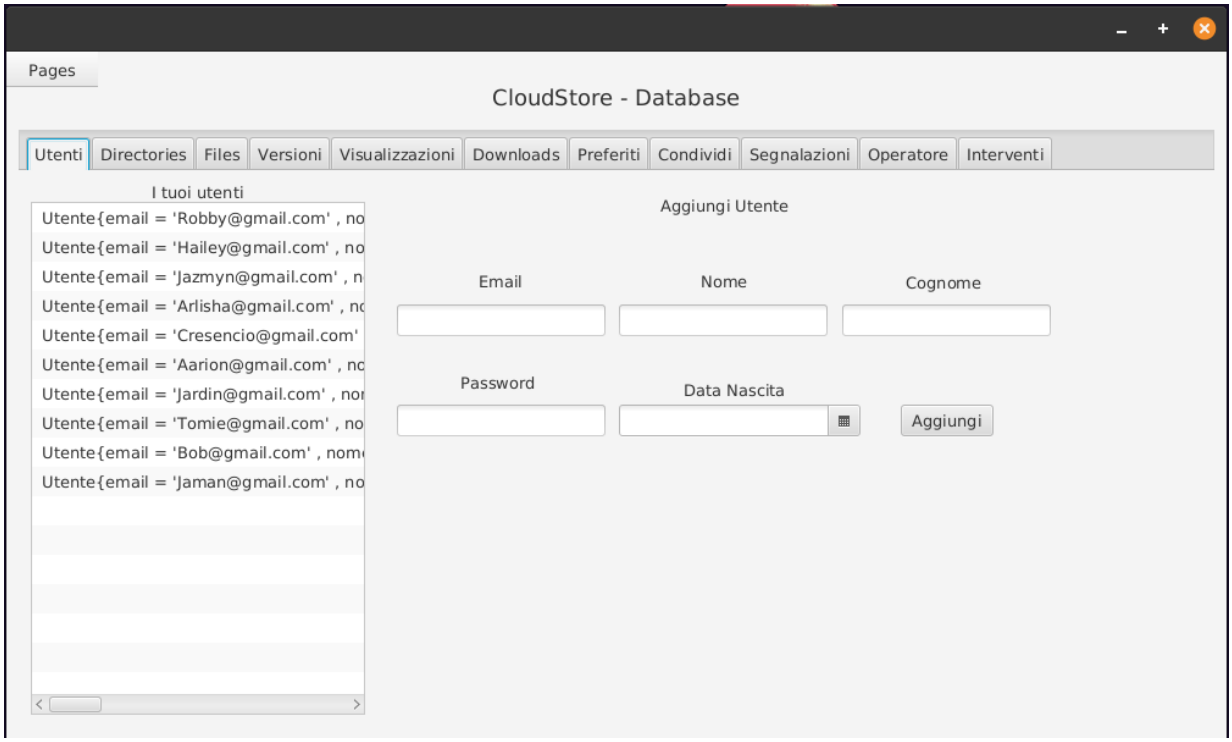

Figura 4.7: Sezione Database dell'applicazione

Unica Tab che ha una particolarità è quelle delle Segnalazioni, dove sulla destra oltre alla possibilità di inserimento di una nuova segnalazioni troviamo anche la possibilità di fare accettare una segnalazione ad un operatore e successivamente di chiuderla.
Per potere effettuare questa operazione andrà prima selezionata una segnalazione dalla lista a sinistra, poi bisognerà scegliere un operatore dal menù a tendina e poi possiamo cliccare su 'Accetta' per fare accettare la segnalazione all'operatore selezionato.
Analogamente per la chiusurà di una segnalazione dovremo selezionarne una a sinistra e poi premere il bottone 'Chiudi'

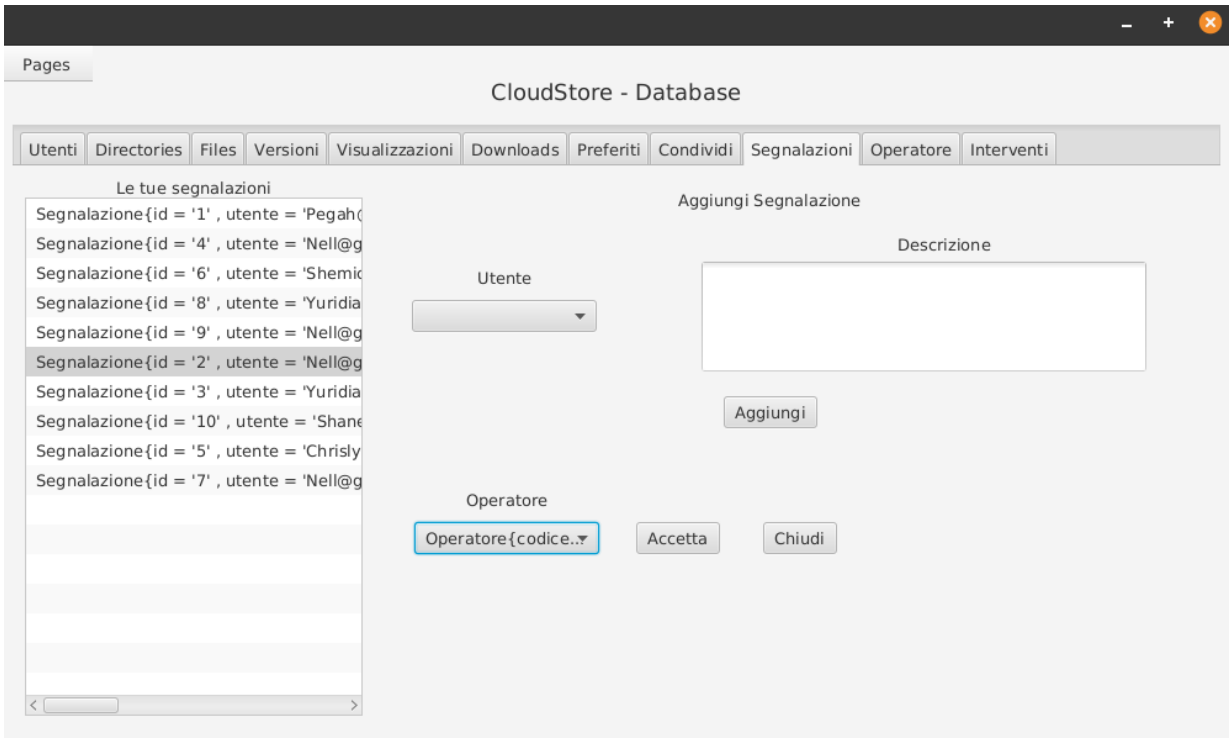

Figura 4.8: Tab dedicata alle segnalazioni

4.2.3 Analisi CloudStore

Questa è la sezione dedicata alle analisi da parte dell'azienda CloudStore.
L'interfaccia è molto semplice e consiste in una lista di operazioni a sinistra mentre a destra abbiamo il risultato delle operazioni.
Per eseguire un operazioni sarà semplicemente necessario selezionare una delle operazioni cliccandoci sopra e automaticamente verrà eseguita la query corrispondente e ne verrà mostrato il risultato nella sezione a destra.

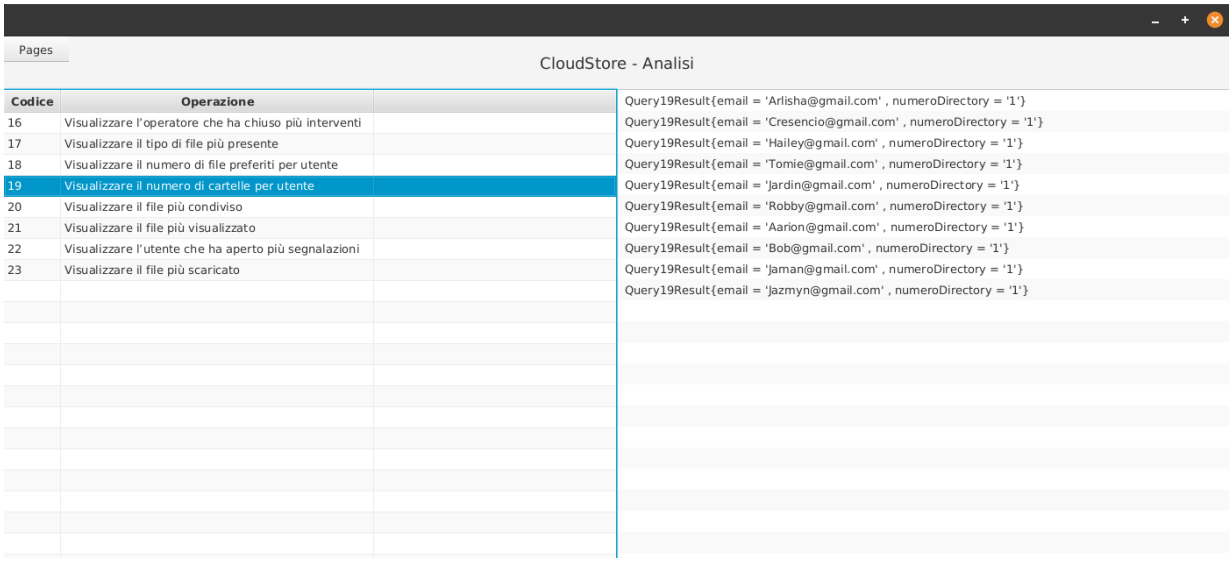

Figura 4.9: Sezione analisi CloudStore dell'applicazione

4.2.4 Analisi Utente

L'ultima sezione dell'applicazione corrisponde alla pagina dedicata alle operazioni specifiche di un utente che in questo caso è generalizzata tramite l'utilizzo delle choicebox che ci permettono di scegliere l'utente, la directory o il file di riferimento della query.
Per quanto riguarda la selezione di un operazione ci troviamo in una situazione analoga alla pagina di analisi precedente dove ci basterà cliccare su una operazione della lista a sinistra ed essa verrà automaticamente eseguita e ne verrà mostrato il risultato nella sezione a destra.

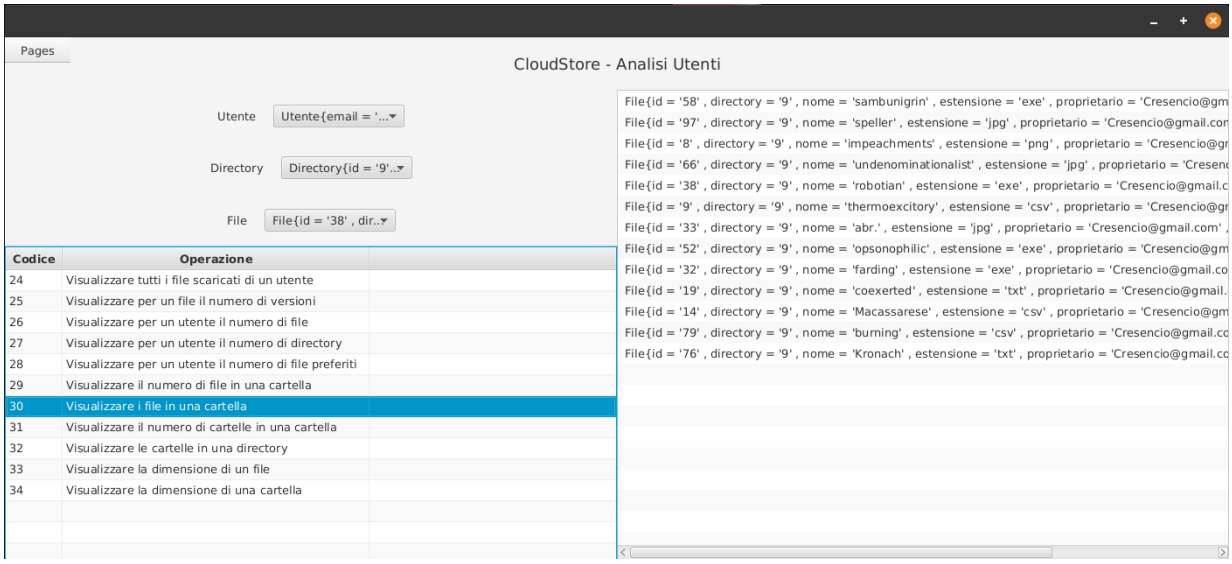

Figura 4.10: Sezione analisi Utente dell'applicazione