**Deel: un’ applicazione per l’ hosting, condivisione e backup di file.**

*Descrizione generale*

Paragrafo 1. Funzionalità

L’ applicazione web da noi sviluppata permette l’ hosting e la condivisione di file di vario genere e di varia dimensione.

L’ utente una volta registrato ed aver effettuato l’ accesso al sistema potrà usufruire delle funzionalità del sito, tra cui : upload di uno o più file sul sistema, la loro condivisione, la possibilità di sfruttare le funzionalità di backup e revisionamento.

Paragrafo 2. Struttura

L’ applicazione è stata sviluppata utilizzando il pattern di programmazione MVC. Tuttavia non tutta la logica riguardante la componente dei controller viene implementata lato server ma, attraverso tecnologie di chiamate asincrone (AJAX), anche lato client.

L’ applicazione è suddivisa in diversi moduli interdipendenti tra loro, ognuno dei quali indirizza problematiche specifiche dell’ applicazione.

Ogni modulo espone solo delle interfacce nascondendo l’ implementazione aumentando la mantenibilità del codice. I principali moduli sono mostrati nella figura in basso.

Domain : Le classi che fanno parte del dominio rappresentano la nostra modellazione degli oggetti reali. Le classi sono strutturate come semplici POJO (plain old java object) così da non dipendere da nessun tipo di framework in particolare.

Data layer : si occupa della persistenza dei dati cioè alla possibilità di salvare in modo permanente dei dati. Per salvare i metadati relativi ai file ed i dati relativi agli utenti è stato utilizzato un DBMS, in particolare MySQL. Per il salvataggio vero e proprio dei file sono state utilizzate le funzionalità del filesystem esposte dal sistema operativo. Per consentire il mapping tra oggetti del database e oggetti utilizzati dal sistema sono state utilizzate le funzionalità del framework Hibernate, utilizzato per realizzare l’ ORM.

Service Layer : si occupa di implementare i principali casi d’ uso dell’ applicazione, manipolando sia oggetti del domain sia del data layer.

Il service layer, come il layer superiore web si avvale di alcune delle funzionalità del framework Spring(in modo particolare “dependency injection” ) in modo di escludere dal codice tutte le logiche di interdipendenza tra le varie implementazioni.

Web Layer : si occupa della logica di navigazione e della gestione della sicurezza. Per la realizzazione di questo layer sono state utilizzate tecnologie di Spring tra le quali Spring MVC e Spring Security.

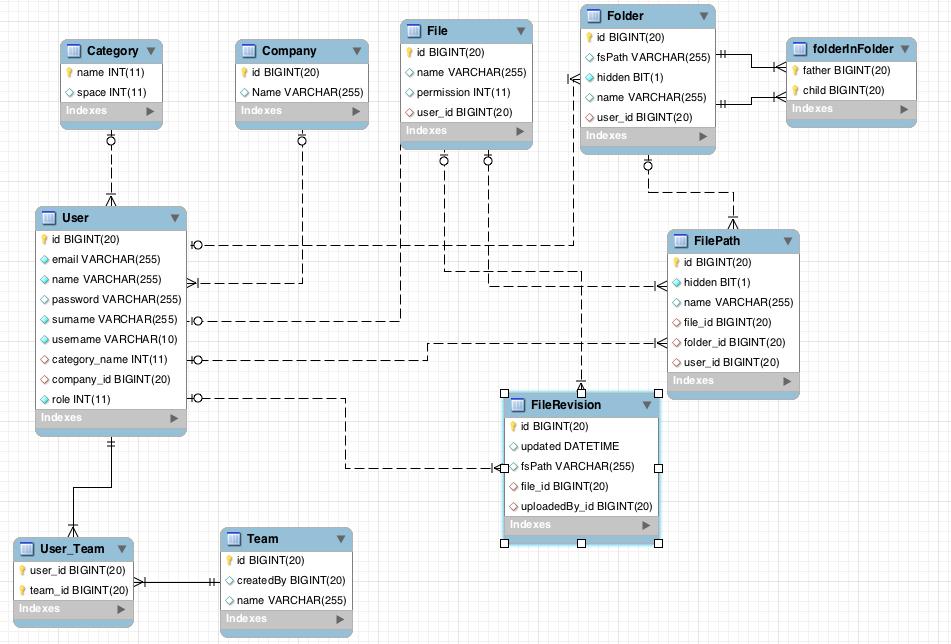
User Interface Layer : è la parte esposta all’ utente e si occupa dell’ interazione con esso. Per lo sviluppo sono state utilizzate le tecnologie

JSTL per quanto riguardo il lato server e jQuery lato client.

Paragrafo 3. DBMS

Per la realizzazione del database è stato utilizzato MySQL. Lo schema E-R dell’ applicazione è definito nella figura in basso.

Per modellare una struttura ad albero rappresentante la struttura organizzativa dei file dell’ utente sono state utilizzate le tabelle Folder e FilePath, dove un filepath appartiene ad ed una sola directory e Folder ha una relazione ricorsiva. Ogni FilePath sarà collegato con un solo File e ciò permette una migliore gestione della condivisione.



Paragrafo 4. ORM

Quasi tutte le relazioni sono state mappate in modo bidirezionale per motivi di efficienza. E’ stato inoltre utilizzato il caricamento Lazy in modo di ridurre i tempi di caricamento ed evitare un catastrofico caricamento ricorsivo di tutto il Database in memoria! Hibernate è stato configurato per essere integrato con Spring. Infatti l’ utilizzo del transactionManager(di Spring) e della sessionFactory verrà fatto attraverso dei Bean e con il frequente utilizzo della Dependency Injection.

Cascade -> Per un controllo maggiore da parte della Business Logic si è evitate di utilizzare inserimenti a cascate. Questo costrutto è stato utilizzato solo in particolari casi che prevedevano una minima propagazione.

Paragrafo 5. Transaction Strategies and Model

Come modello di transazione è stato utilizzato il modello dichiarativo che consente di non mischiare codice relativo alla logica del programma con codice relativo all’ accesso ai dati nonché meno error prone.

In questo modello il contenitore(nel nostro caso Spring attraverso Hibernate) si occupa di iniziare, committare e rollbackare le transazioni.

Come strategie invece abbiamo definito come Unit Of Work nella nostra applicazione un intero caso d’ uso, implementato come metodo nel modulo Service. Utilizzando quindi l’ annotazione @Transactional su questi metodi abbiamo ottenuto una non intrusiva e consistente strategia di transazione.

Paragrafo 6. Web ed User Interface

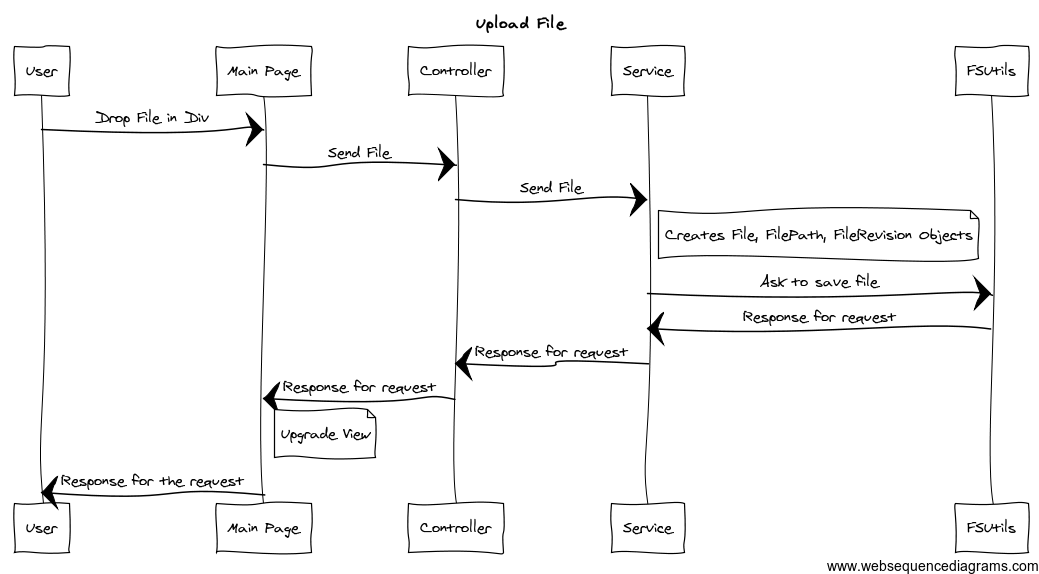
Per migliorare l’ interfaccia utente si è scelto di mantenere le funzionalità usate più frequentemente sulla stessa pagina facendo uso di chiamate asincrone presenti nelle librerie jQuery. Per questo la maggior parte della logica di navigazione riguarda quindi casi d’ uso meno frequenti come la registrazione, il login o l’ interfaccia amministrazione. Per quanto riguarda autenticazione ed access control è stato usato un modulo Spring denominato Spring Security che tramite filtri servlet gestisce l’ accesso alle pagine ed il login. Quest’ ultimo modulo è configurabile quasi totalmente tramite una pagina xml nella quale vengono fatti dei settaggi.

Paragrafo 7. Note sui test

Sono state adottate molte tecniche di sviluppo in base ai moduli implementati tra le quali Test-driven Development utilizzando i popolari framework di testing JUnit e Mockito.

Paragrafo 8. Esempio di un caso d’ uso

In questo paragrafo verrà descritto un caso d’ uso d’ esempio.



Per l’ upload di un file l’ utente quindi trascinerà il file nel div dopodiché verrà mandata una richiesta al controller che tramite il modulo service gestirà l’ inserimento o il revisionamento del file. Andata a buon fine la transazione la pagina tramite una callback aggiornerà la vista dinamicamente senza ricaricare la pagina.

Paragrafo 9. Conclusioni

Uno dei vantaggi di aver utilizzato Spring ed in particolar modo la dependency injection è stato quello di aver reso per quanto più possibile indipendenti i moduli del sistema che non dovendo conoscere nessun dettaglio implementativo possono essere sia migliorati che testati singolarmente.