Manuale di installazione di “Biblioteca di risorse”

Installazione ed utilizzo.

Per utilizzare questo software, sono necessari i seguenti passaggi:

1. Scompattate la directory “biblioteca_quinta_release” nel vostro Desktop;
2. Successivamente, aprite un prompt dei comandi se siete su windows, o una shell interattiva se siete su Linux o Mac(bash, sh, iTerm, e similari);
3. Dirigetevi da shell nella directory contenente “biblioteca_quinta_release”;
4. Entrate dentro “biblioteca_quinta_release”;
5. Entrate dentro la cartella “dist”;
6. Incollare questo comando: java -jar "biblioteca_quinta_release.jar" (doppi apici compresi);
    a. Per ogni release è presente la corrispondente release di debug, che contiene tutto l’output di debug che concerne le query eseguite dal database; per avviarle, in caso potesse interessarne l’esecuzione, basterà digitare:

       java -jar “nome_programma_DEBUG.jar”, doppi apici compresi.

7. Il vostro programma verrà eseguito; non avete bisogno di librerie esterne, in quanto tutte le risorse esterne necessarie sono contenute dentro la cartella /dist/lib;
8. Attenzione a non rimuovere il file “categories.txt”; contiene tutte le categorie presenti nel vostro software. Senza di esso, il programma non potrà funzionare.

Interpretazione dei requisiti funzionali, e non funzionali.

Nello sviluppare questo software, sono state fatte queste supposizioni, dal punto di vista progettuale ed implementativo:

1. Abbiamo utilizzato un database management system locale, ovvero SQLite, per gestire la nostra base di dati; questa scelta è stata fatta in quanto:
    a. Secondo noi, era molto più coerente per il principio del riutilizzo delle tecnologie già create, di riutilizzare qualcosa di efficiente, testato e funzionante, che implementare questa funzionalità noi stessi.

    b. Vedevamo come una complicazione onerosa dover gestire ed implementare tutta la problematica di salvataggio, serializzazione, organizzazione, strutturazione di accesso, nonchè sincronizzazione, mediante l'utilizzo di semplice sincronizzazione di files di testo, e quindi l'utilizzo di un dbms locale ci è apparsa per il nostro contesto e necessità, una scelta ottimale.
    c. Il dbms ci permette una libertà impareggiabile nel poter memorizzare, recuperare ed elaborare i dati della applicazione; di contro, richiede una progettazione più curata a livello di ingegneria delle classi che devono gestire il recupero ed il salvataggio dei dati da database entro le classi rappresentanti i modelli delle entità(per citare un esempio, quello che fanno i framework di sviluppo MVC che offrono accesso ad un utilizzo di dbms). Questa parte dell'applicazione verrà sviluppata in meglio nella seconda parte del progetto, dove apporteremo l'utilizzo e l'implementazione di patterns appropriati per gestire in maniera mirata e dedicata le responsabilità tra le classi.
2. L'utilizzo di un dbms permette con molta più facilità, visto l'utilizzo quasi nativo di mySQL, di:
    a. Poter rappresentare in maniera agile e rapida l'inserimento di nuove strutture dati in locale utilizzabili per memorizzare qualsiasi tipo di dato inerente a risorse ed altro(supporta anche campi di dato di tipo BLOB, utili per rappresentare a livello di byte i dati, e si può utilizzare in maniera praticamente libera per rappresentare qualsiasi cosa, entro i limiti fisici e del database ovviamente).
    b. Si ha uno strumento molto potente per poter elaborare ed analizzare i dati contenuti all'interno del database, mediante interrogazioni con query annidate ed elaborate, con la maggior parte degli operatori disponibili su mySQL.
3. Le categorie sono gestite simulando la directory di un file system, all'interno del file "categories.txt"; le categorie si dividono in 3 parti, all'interno del file:
    a. Categorie padre senza figli (10 parole separate da spazi);
    b. Categorie padre con figli (9 parole separate da spazi);
    c. Categorie figlie, con un relativo padre(11 parole separate da spazi).
    d. Questa decisione sulle categorie è stata effettuata per avere un metodo semplice per poter rispecchiare un files system; La rappresentazione delle categorie è indipendente dalla implementazione effettiva del files system, quindi basta cambiare il metodo appropriato che si occupa di memorizzare le categorie, per poter completamente utilizzare un'altra metodologia per poter rappresentare il files system, piuttosto che implementare una lettura da remoto o una lettura locale sul files system della macchina ospitante l'applicazione. Questa scelta è stata effettuata sia per mantenere e concentrare gli sforzi su punti più focali dei requisiti funzionali, che per avere un unico punto di accesso all'implementazione di questo requisito dell'applicazione.
    e. Ogni Categoria contiene un suo identificativo testuale univoco, che rappresenta univocamente la categoria, e viene utilizzata da ogni oggetto per riferirsi ad essa o indicarne l'appartenenza a tale categoria(ogni categoria, se non ha figli ha solo una risorsa, mentre se ha sottocategoria, ogni sottocategoria ha una sola risorsa contenutavi al proprio interno).

4. Ogni utente viene registrato con l'username che indica; tale username è univoco per ogni utente, e quindi non possono esistere due utenti con un univoco username.
5. Relativamente all'aggiunta degli Operatori, abbiamo deciso che solamente un Operatore può aggiungere un altro Operatore. Ma in questo modo è sorto un altro problema, ovvero come poter aggiungere un Operatore se all'avvio del programma non siano disponibili Operatori registrati. La nostra soluzione è stata quella di creare, fin dal primo avvio del programma, un Operatore di default, identificato dall'username "default" e password "default". Effettuando il login con queste credenziali, sarà possibile aggiungere un altro Operatore. L'idea è che al primo avvio ci si identifichi com questo profilo, per aggiungere il primo Operatore reale dell'applicazione. Infine, ogni Operatore può creare un altro Operatore, ma sono tutti dotati di stesso livello di gerarchia e autorizzazioni.
6. Dentro ogni release, vi è contenuto al suo interno il diagramma delle tabelle implementato nel database, rappresentante dove l'applicazione ha salvato le proprie tabelle e su che struttura dati.
7. Per attivare l'output di debug, basta entrare nella classe Menu, ed impostare la costante "private static final boolean DEBUG" a true; serve ricompilare il programma. Come esempio, viene rilasciata la quarta release sia in eseguibile di debug, che in eseguibile di prova.