Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione
Corso di Laurea in Ingegneria Informatica

Progettazione e realizzazione di moduli software per la gestione di posta elettronica e trasferimento dati

Laureando:
Matteo Bassani

Relatore:
Ch.mo Prof. Giorgio Maria Di Nunzio

Anno accademico 2013/2014

A Ilaria, i miei genitori e Nicolò

Indice

Capitolo 1

Introduzione

La tesi riguarda il lavoro svolto durante il tirocinio presso l'azienda Zero12 s.r.l. con sede operativa a Carmignano di Brenta. Lo stage ha avuto una durata di due settimane per un totale di 320 ore dal 2 settembre al 25 ottobre 2013.

Lo scopo della tesi è stato quello di acquisire le competenze di studio, progettazione e sviluppo di due moduli software necessari all'evoluzione del sistema DriveFarm: Gmail plugin e Data Importer.

Per la loro realizzazione sono stati utilizzati i linguaggi xml, html, javascript, python.

Il primo modulo ha l'obiettivo di facilitare l'utente, o azienda, nell'includere documenti presenti nel proprio profilo Drivefarm in un messaggio di posta elettronica. Dato il continuo incremento dell'utilizzo del client di posta Gmail, per fare questo si è utilizzato uno strumento messo a disposizione dallo stesso client: il gadget. Una volta installato, l'utente ha la possibilità di visionare i propri contenuti di profilo Drivefarm in una struttura ad albero. Per ogni contenuto presente è possibile eseguire le seguenti azioni: salvare il link al file in una mail, scaricare il documento, renderlo pubblico.

Con il secondo modulo verrà data la possibilità di trasferire i documenti desiderati dal sistema di archiviazione Dropbox al sistema Drivefarm. Per far questo si utilizza un programma sviluppato in Python con il quale, mediante una serie di passaggi, è possibile visionare i propri file presenti in Dropbox e indicare la cartella di Drivefarm in cui si intende compiere la migrazione.

Le due applicazioni permettono una maggiore visibilità al File Server Drivefarm. Esistono già infatti molti concorrenti affermati come Dropbox, Google Drive o SkyDrive che sfruttano la tecnologia Cloud per l'archiviazione dei dati. Grazie ai due moduli sarà possibile semplificare alcune operazioni, rendendo l'utilizzo di Drivefarm più semplice e comodo.

Capitolo 2

Progettazione

2.1 Drivefarm

I due moduli fanno riferimento al sistema di archiviazione dati Drivefarm[1], sviluppato dall'azienda presso cui è stato svolto il tirocinio.

2.1.1 Descrizione

Drivefarm è un File Server, basato su tecnologia Cloud, progettato e sviluppato da Zero12 s.r.l. con lo scopo di rispondere alle esigenze aziendali di conservazione ed accesso al patrimonio documentale di una ditta.

In particolare, col termine File Server, si intende una macchina progettata per mettere a disposizione degli utilizzatori di una rete dello spazio su disco, nel quale sia possibile salvare, leggere, modificare, creare file e cartelle condivise con tutti.

Caratteristiche di Drivefarm sono: la semplicità, è infatti possibile per ogni componente aziendale disporre di un proprio spazio su Cloud dove conservare molteplici documenti di lavoro ed avere accesso agli stessi da ogni postazione fissa e mobile dotata di una connessione a internet.

Presenta svariati livelli di sicurezza: dal punto di vista infrastrutturale è basata su Amazon Web Services[2], da quello applicativo garantiscono le connessioni HTTPS su cui si basa il trasferimento delle informazioni e da quello amministrativo, offre possibilità agli utenti amministratori di gestire i privilegi di accesso al sistema dei singoli utenti, limitando alcune funzionalità se necessario.

La struttura Drivefarm fornisce inoltre un sistema di log per gestire le attività condivisioni dei file anche al di fuori dei sistemi aziendali in modo tale da identificare un eventuale utilizzo improprio di documenti riservati.

2.2 Gadget Gmail

Il primo modulo sviluppato durante il tirocinio riguarda la realizzazione di un Gadget Gmail. Si tratta di una mini applicazione scritta in XML con all'interno del codice

[1] https://www.drivefarm.com/
[2] http://aws.amazon.com/

HTML, Javascript, Flash o Silverlight. Ogni gadget è ideato per funzionare su più siti e piattaforme, perciò vi sono tag speciali e librerie che agiscono in luoghi diversi.

2.2.1 Concetti base

La prima riga contiene il codice standard per avviare un file di tipo XML. Seguono poi alcuni tag che indicano la presenza di un gadget all'interno. Questi tag sono:

- `<Module>`

  che indica la presenza di un gadget;

- `<ModulePrefs>`

  che contiene informazioni come il titolo, la descrizione, l'autore e altre funzioni opzionali;

- `<Content type=‘‘...’’>`

  che indica il tipo di contenuto del gadget (HTML o url);

- `<! [CDATA [...  ... ]]>`

  usato per racchiudere il codice funzionante. Tali parentesi indicano che non dev'essere usato il linguaggio XML all'interno, ma solo HTML o Javascript;

- `</ Content>`

  che indica la fine della sezione contenuti;

- `</ Module>`

  che significa la fine del gadget.

Esempio di gadget è il seguente, la cui funzionalità è di far comparire la scritta "Hello, World!" nell'area riservatagli.

```
<?xml version=‘‘1.0’’ encoding=‘‘UTF-8’’ ?>
<Module>
  <ModulePrefs title=‘‘hello world exemple’’ />
  <Content type=‘‘html’’>
    <![CDATA[Hello, World!]]>
  </Content>
</Module>
```

2.2.2 Sidebar Gadget in Gmail

Nella finestra di Gmail è possibile inserire dei gadget. Tali applicazioni prendono il nome di Sidebar Gadget e vengono poste nella sezione in basso a sinistra, resa visibile al click del pulsante con tre puntini ("..."), presente nel fondo della pagina.

La procedura per l'inserimento di un nuovo gadget consiste nel:

- pubblicare un gadget in una locazione raggiungibile da internet;
- attivare la lab "Aggiungi un gadget tramite URL" dalla sezione Labs delle impostazioni di Gmal;
- verrà creata una nuova sezione denominata "Gadget" da cui è possibile inserire l'Url che riferisce al proprio gadget;
- una volta premuto su "Aggiungi", il gadget verrà visualizzato.

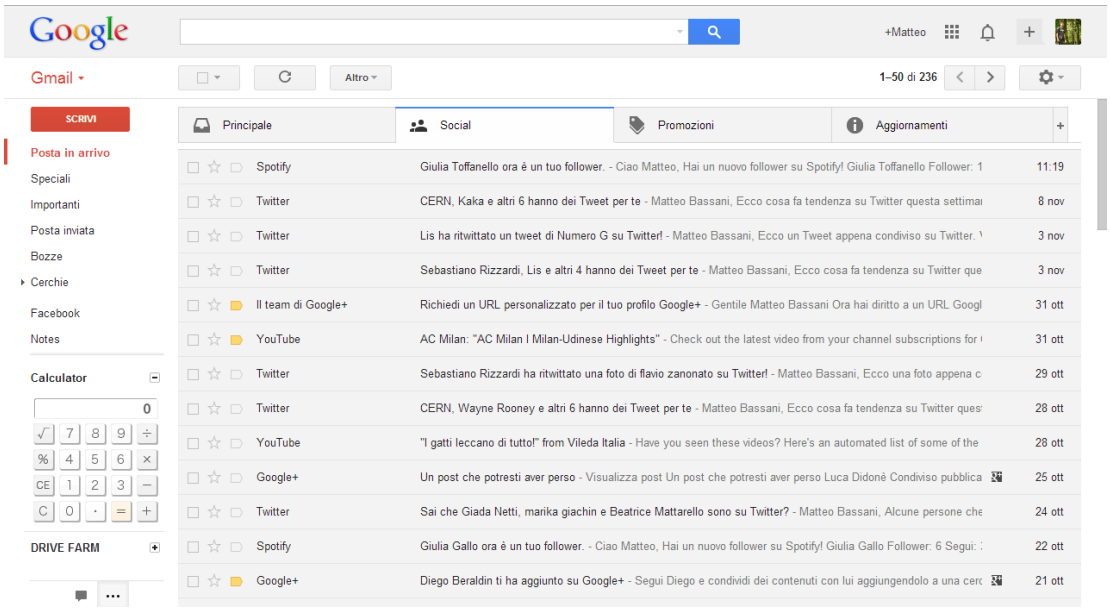

Figura 2.1: Esempio di un gadget con funzione di calcolatrice

2.3 Ambiente di sviluppo

2.3.1 Google Gadget Editor

Google fornisce un proprio editor per la realizzazione di gadget. Tale editor è anch'esso un gadget, da inserire nella propria pagina iGoogle.

Quest'editor presenta varie funzioni come salvare file xml, modificarli e pubblicarli in un server Google, dal quale è possibile recuperare l'Url, per testare il proprio gadget.

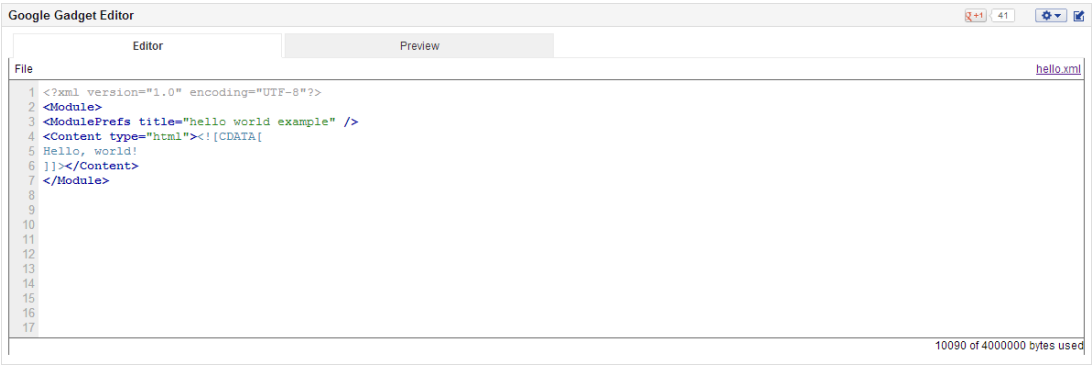

Figura 2.2: Google Gadget Editor

2.3.2 Eclipse e Aptana

Il tag "Content type" oltre ad "html" supporta anche "url". Tale funzione permette di convertire un intero sito in un gadget. Un ambiente di sviluppo integrato, adatto per sviluppare applicazioni web è Eclipse con il plug-in Aptana.

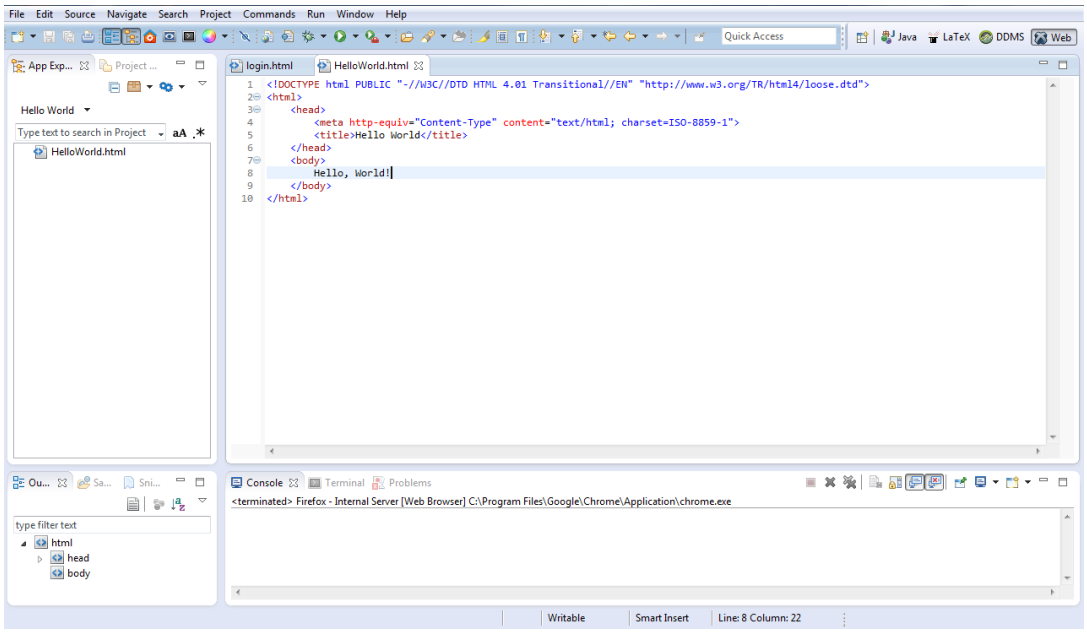

Figura 2.3: Eclipse

Grazie ad esso, tra le tante funzionalità è possibile comporre file html, css, JavaScript e testarli sul proprio browser premendo il pulsante run.

Per la realizzazione del primo modulo sono stati usati entrambi gli ambienti di sviluppo.

Per quanto riguarda il secondo progetto è stato utilizzato lo stesso Eclipse ma con una differente "Perspective". Il plug-in Aptana, infatti, permette di installare l'estensione "PyDev", con la quale si gestiscono i progetti nel linguaggio "Python".

Capitolo 3

Realizzazione

3.1 Gmail plugin

3.1.1 Descrizione

Questo modulo è composto da un sidebar gadget per Gmail, capace di comunicare con il progetto Drivefarm. Il gadget si sviluppa in due finestre. La prima rappresenta la schermata di login dove vi sono impressi il logo di Drivefarm, due Editbox per l'inserimento di username e password, un Pushbutton per la "login" e un link al sito ufficiale di Drivefarm per gli utenti che non conoscessero tale servizio. La seconda schermata ha una prima sezione dedicata alla visualizzazione dei documenti presenti dall'utente Drivefarm sotto forma di albero e un Pushbutton sottostante per il "logout".

3.1.2 Login.html

Figura 3.1: Schermata di login

body

L'html della prima schermata, alla sezione "body", contiene un tag "form" per l'accesso al profilo Drivefarm. In tale "form" sono racchiusi tre tag input per l'inserimento di mail e password e un link al sito ufficiale. Per completare il layout della schermata è stata posizionata un'immagine del logo Drivefarm. Al click del pulsante "login" viene eseguita la funzione "recuperoDati" descritta nella sezione "head".

head

La funzione "recuperoDati" presente gestisce i vari errori che l'utente può compiere nell'effettuare l'accesso. Il primo controllo riguarda il mancato inserimento della mail, il secondo quello della password. In entrambi i casi l'utente viene allertato con un "alert".

Nel terzo controllo si verifica che la coppia username e password esistano nei registri Drivefarm. Viene quindi inviata una richiesta codice di autorizzazione xauth, per ottenere l'accessToken. Tale codice permette di eseguire tutte le chiamate get e post per avere accesso ai dati dell'utente. Qualora la risposta a questa chiamata fosse un errore viene segnalato all'utente tramite un "alert". Se invece la chiamata va a buon fine, l'accessToken viene passato alla schermata successiva.

3.1.3 Tree.html

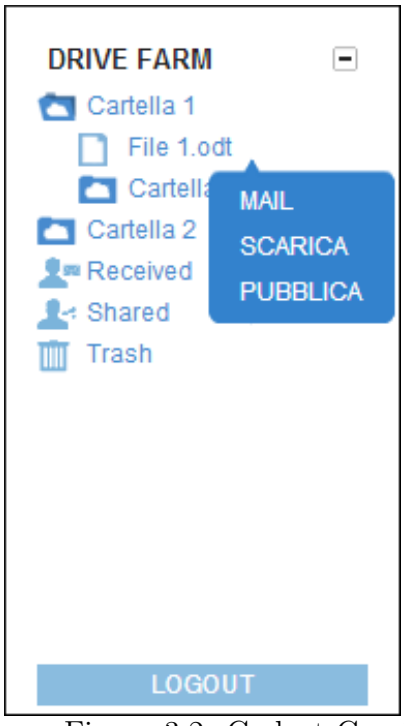

Figura 3.2: Gadget Gmail

head

Il file "Tree.html" è il cuore del progetto: contiene e visualizza la strutta ad albero composta dai file presenti nell'account Drivefarm. Tale sorgente è composto da cinque funzioni javascript nella sua sezione "head".

La prima si chiama "getParameter()" e, come suggerisce il nome, permette di ottenere il parametro passato nell'url dalla pagina precedente. Tale parametro è l'accessToken dell'utente che ha effettuato l'accesso. Grazie ad esso sarà possibile effettuare tutte le chiamate get e post verso Drivefarm.

La seconda funzione è "jsonByGet()" che riceve come parametro una path a una cartella presente nel profilo Drivefarm dell'utente e restituisce il contenuto in formato stringa. La funzione agisce secondo quanto segue: viene invocata "getJSON", funzione di jQuery (framework javascript), alla quale è passata una stringa avente l'url di accesso a Drivefarm, l'accessToken e la path. In caso di successo, viene restituito il parametro "data", opportunamente convertito dal formato "json" in stringa per essere analizzato. "data" contiene tutte le informazioni riguardo a file e directory contenuti dentro la cartella indicata nella path.

Viene quindi invocata la terza funzione: "parseJson()". Lo scopo di questa funzione è organizzare i dati letti dalla stringa data ricevuta. In un array chiamato "contenuto" vengono inseriti la path di ogni dato e un booleano. True se il dato rappresenta una cartella, false se si tratta di un file.

"makeBrach()" è la quarta funzione. Questa scrive in linguaggio html la porzione di albero ricavata dalle funzioni precedenti. Ovvero, per ogni dato presente nell'array "contenuto", associa l'immagine corretta (determinata dall'estensione del file o dal fatto che sia o meno una cartella) e lo colloca al posto giusto nell'albero. In "makeBrach()" viene inoltre gestito un contextMenu, sempre con l'ausilio di jQuery. Si tratta di un menù a tendina che compare quando un file viene cliccato con il tasto destro del mouse. È riservato solo ai file e non alla cartelle. Grazie a questo menù è possibile eseguire tre azioni quali inserire il link al file in una mail, scaricare il file e renderlo pubblico.

Se l'utente sceglie una opzione presente nel contextMenu viene invocata la funzione "azione()". In essa viene prima di tutto pubblicato il file selezionato tramite una funzione post di jQuery. Poi, in base alla scelta dell'utente, viene eseguita l'azione desiderata.

body

Le funzioni descritte vengono invocate ognuna dalla precedente. La prima invece è invocata dal body del documento html.

Al primo accesso nella pagina, perciò, viene chiamata la funzione "getParameter()", la quale invoca la seconda funzione, "jsonByGet()", dandole come path iniziale la root ("\"). Questo permette di recuperare i file e le cartelle presenti nel livello più alto di Drivefarm e visualizzarli in un tag "div" con "id = tree". L'utente a questo punto potrà vedere il primo livello dell'albero.

Cliccando su una cartella sarà richiamata nuovamente la funzione "jsonByGet()" ma questa volta la path assegnata non sarà più "\" ma il percorso della cartella selezionata. In questo modo potrà essere visualizzato il contenuto della cartella cliccata.

3.2 Data Importer

3.2.1 Descrizione

Il modulo è scritto interamente in linguaggio python. Consiste in un programma a schermate, dove in ognuna di esse è richiesta un'azione da eseguire. Lo stile dell'applicazione è molto semplice e intuitivo.

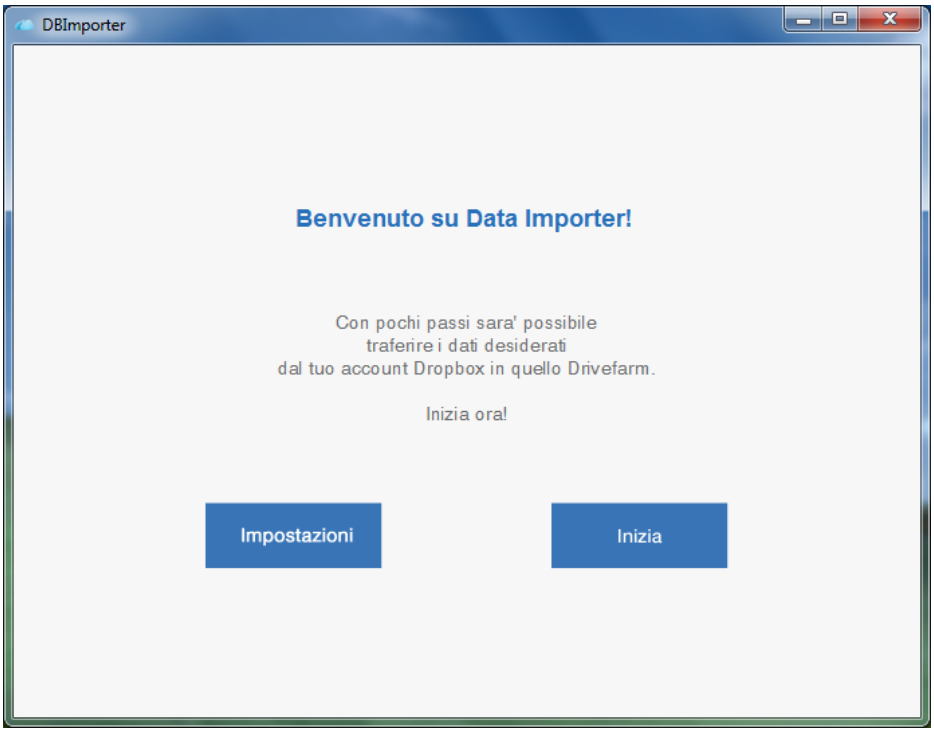

Figura 3.3: Welcome

La schermata di benvenuto (Figura 4.1) accoglie l'utente e spiega brevemente cosa andrà a fare.

Cliccando sul pulsante "inizia" verrà visualizzata una finestra di dialogo più grande (Figura 4.2) in cui sono richieste le credenziali di accesso a Dropbox. Inserite, si chiederà all'utente di permettere al programma di leggere e modificare qualsiasi file. Una volta acconsentito, la finestra di dialogo sarà chiusa. In caso l'utente non acconsenta o chiuda la finestra il programma non procederà, ma comparirà la schermata iniziale.

Se invece il procedimento andasse a buon fine, la finestra si chiuderebbe automaticamente e verrebbe mostrata una schermata di caricamento dei dati (Figura 4.3). Tale schermata ha una progressbar per indicare all'utente che l'azione di caricamento non è immediata ma durerà qualche secondo.

Terminato il caricamento apparirà la schermata mostrata in figura 4.4 in cui sono visualizzati i dati presenti nell'account di Dropbox in una struttura ad albero. Vicino ad ogni dato è presente una checkbox per selezionarlo. Inoltre, cliccando una cartella saranno selezionati in automatico anche tutti i file contenuti all'interno, con la possibilità, però, di deselezionare quelli non desiderati. Una volta scelti i dati che si desida trasferire non resta che premere il button "avanti".

Figura 3.4: Dropbox login

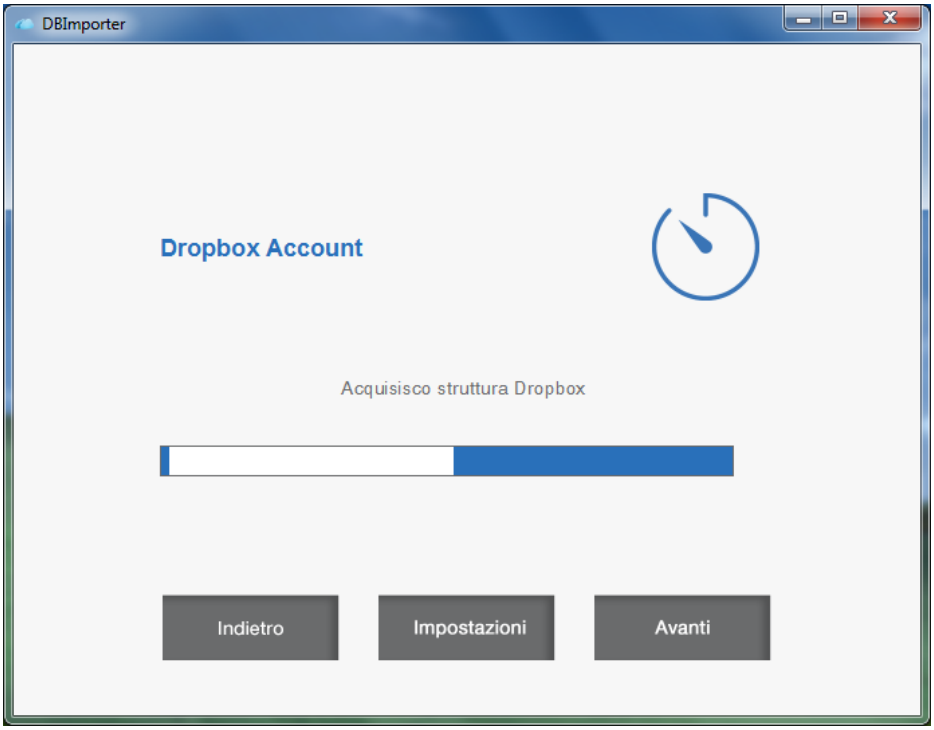

Figura 3.5: Caricamento Dropbox tree

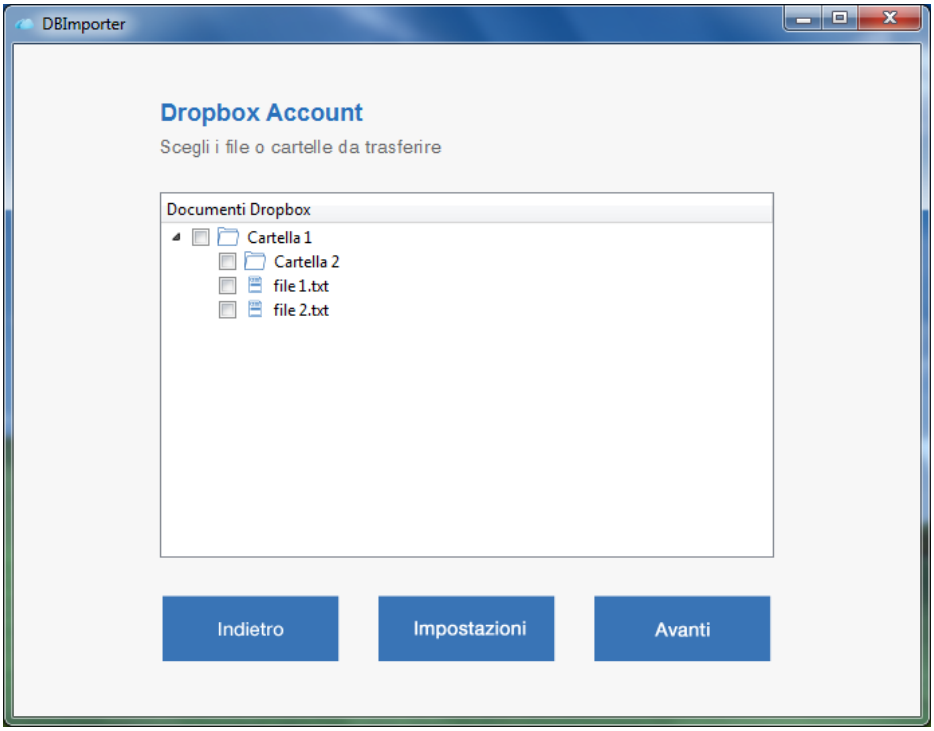

Figura 3.6: Dropbox tree

Il programma controlla che i dati selezionati siano maggiori o uguali a uno, se ciò non dovesse accadere comparirà un alert e non sarà consentito procedere.

La schermata di figura 4.5 permette di inserire le proprie credenziali di Drivefarm. Se saranno mancanti o non corrette l'applicazione non procedrà.

Inserite le credenziali e premuto sul button "avanti" comparirà una finestra di caricamento identica a quella già visualizzata per Dropbox (Figura 4.6).

Dopo qualche secondo la schermata di caricamento lascerà il posto alla visualizzazione in struttura ad albero dei dati presenti in Drivefarm (Figura 4.7). A questo punto sarà possibile scegliere una sola cartella nella quale si desidera importare tutti i file di Dropbox precedentemente selezionati.

Scelta la cartella si accederà ad un'altra schermata di caricamento (Figura 4.8). In questa fase vengono effettutati gli ultimi controlli prima di iniziare il trasferimento. Tali controlli comprendono:

- il calcolo della dimensione dei dati trasferire;
- il calcolo dello spazio libero presente nel computer; i dati verranno prima scaricati nel computer in una cartella temporanea e poi trasferiti in Drivefarm perciò è necessario che lo spazio disponibile nel computer sia maggiore della dimensione totale dei file;
- il calcolo dello spazio libero in Drivefarm; se la dimensione dei dati da trasferire supera lo spazio libero in Drivefarm non viene autorizzato il trasferimento.

Figura 4.9 mostra la schermata per la pubblicazione dei calcoli eseguiti.

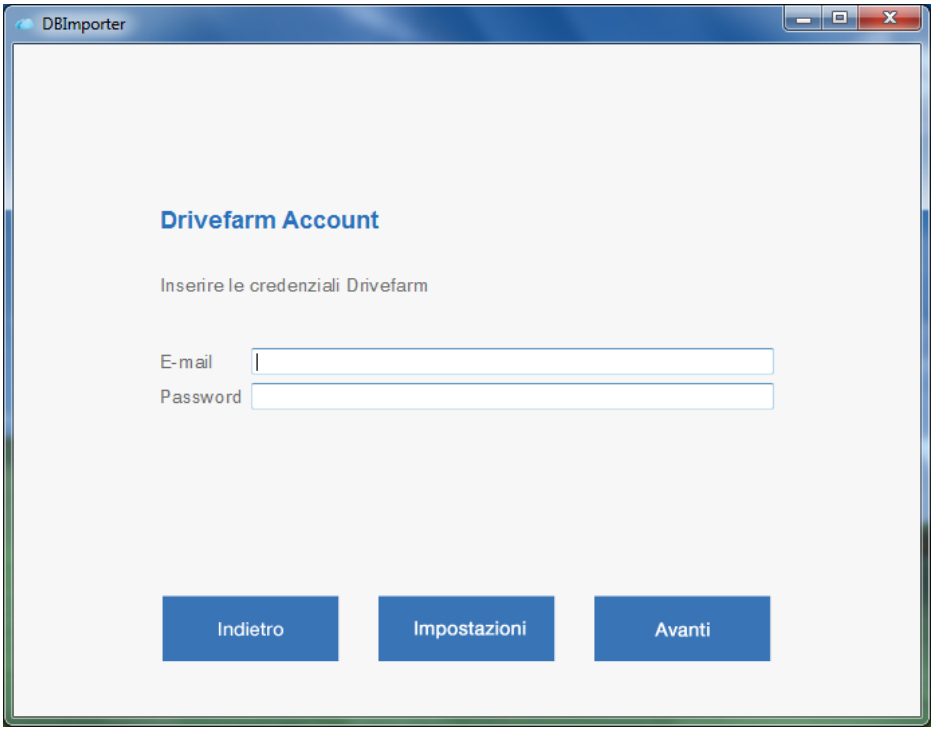

Figura 3.7: Drivefarm login

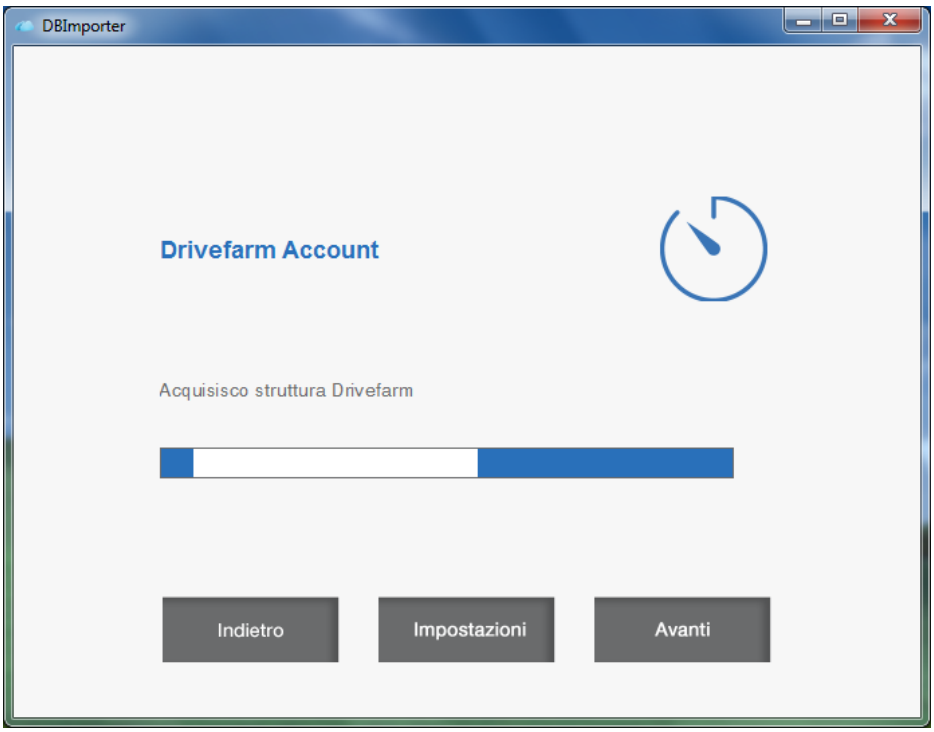

Figura 3.8: Caricamento Drivefarm tree

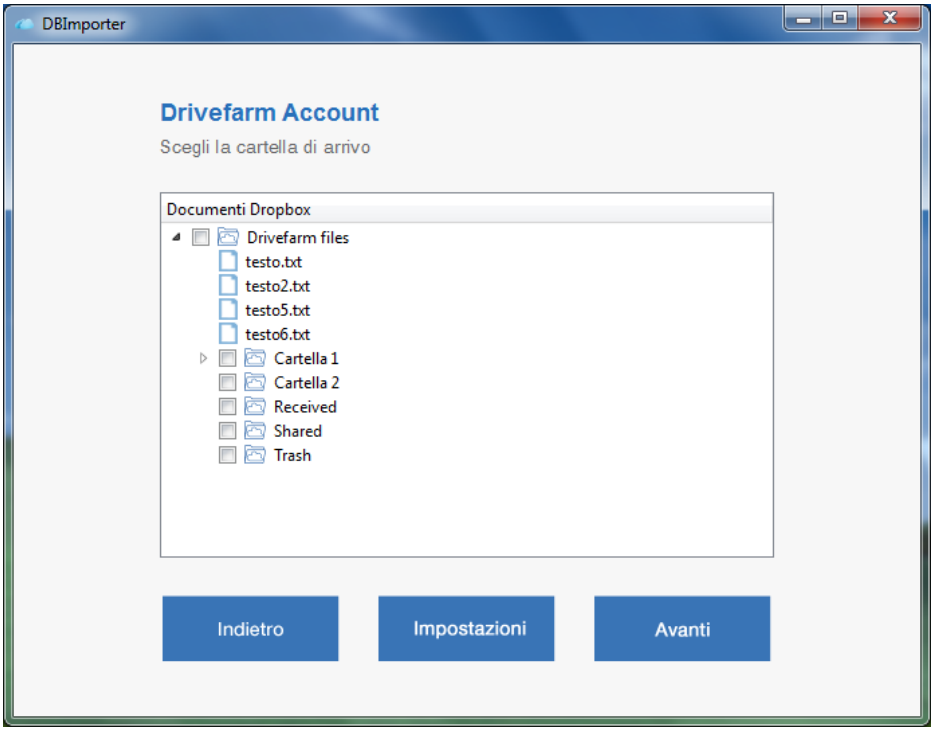

Figura 3.9: Drivefarm tree

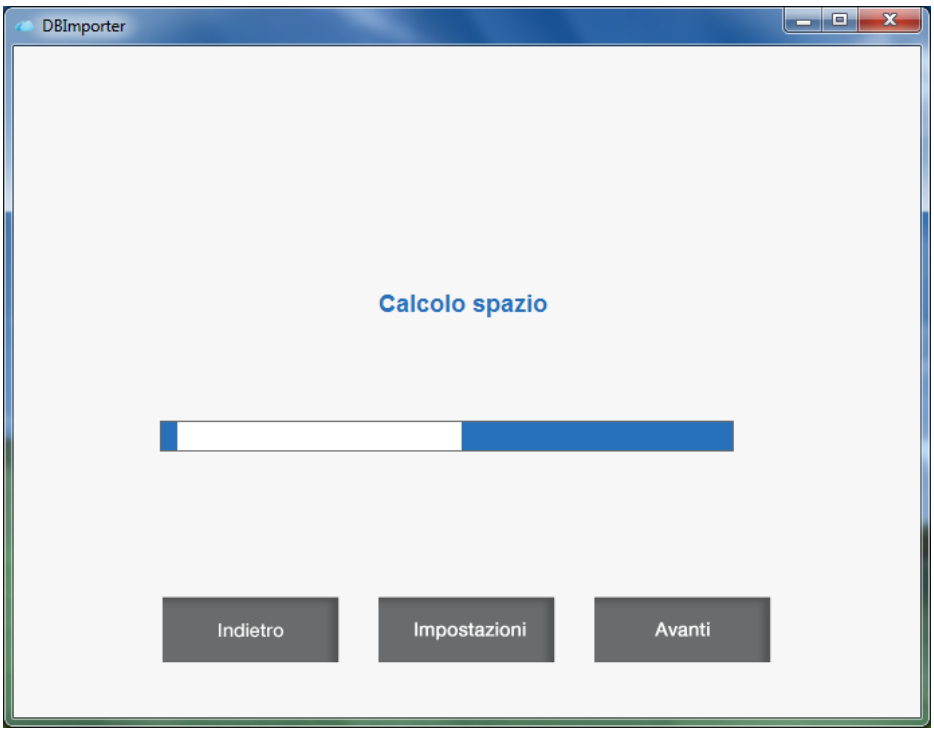

Figura 3.10: Calcolo dimensione

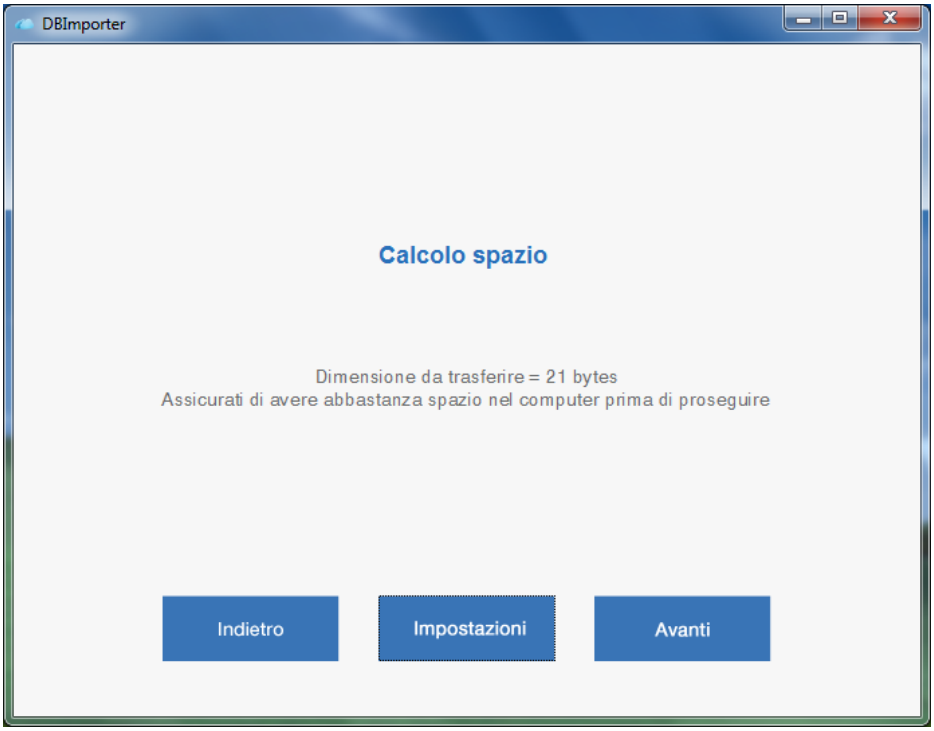

Figura 3.11: Gestione spazio

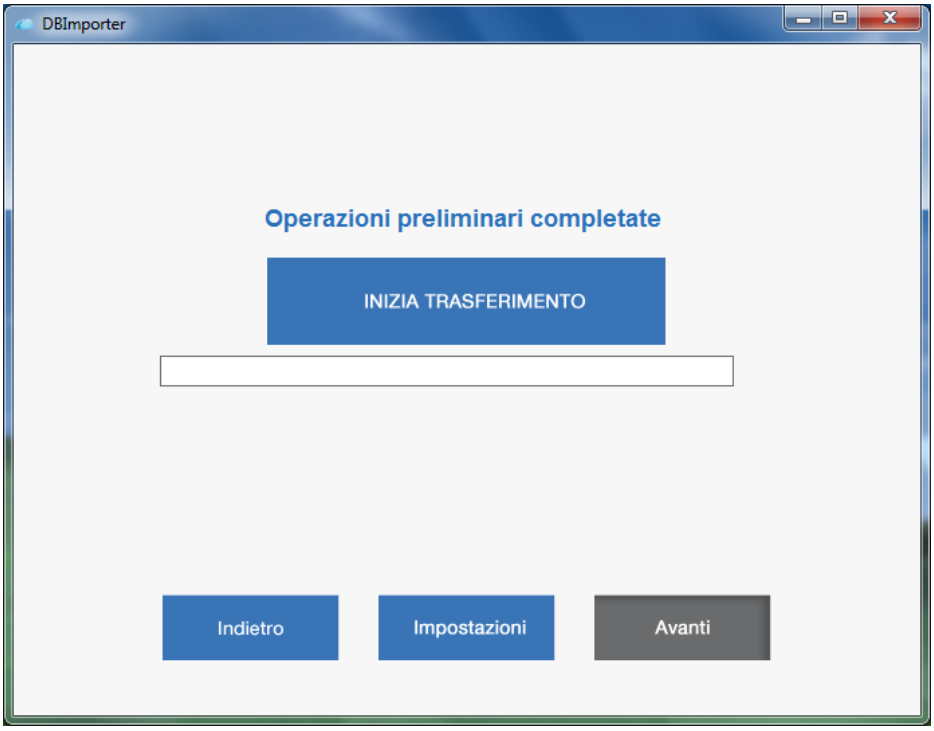

Figura 3.12: Trasferimento

Nell'ultima schermata (Figura 4.10) sono presenti una progressbar in pausa e un button per iniziare la migrazione. Cliccando su quest'ultimo la progressbar comincerà a riempirsi. Sotto di essa compariranno uno alla volta i nomi dei file trasferiti. In qualunque momento è possibile mettere in pausa il trasferimento. Se ciò accade comparirà una finestra di dialogo che chiederà all'utente l'azione desiderata tra: riprendere il trasferimento, saltare il file corrente o annullare tutta l'operazione ed eliminare quindi i file già trasferiti in Drivefarm.

Altra azione richiesta all'utente è il comportamento da tenere quando viene individuato un file o una cartella con lo stesso nome nel profilo Drivefarm. L'utente potrà scegliere tra rinominare il file, saltarlo o sovrascriverlo. Al termine del trasferimento sarà possibile ritornare alla schermata principale premendo il button "avanti" che risulterà cliccabile solo alla fine del processo.

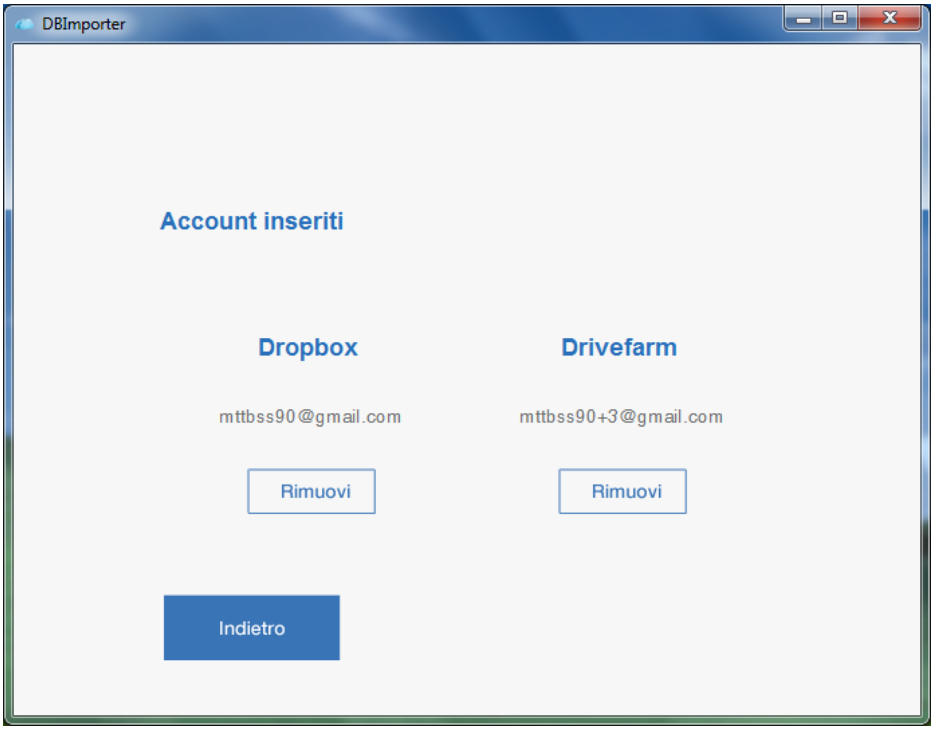

Figura 3.13: Gestione Account

In ogni schermata del programma è presente un button con scritto "impostazioni". La sua presenza è data dal fatto che ogni volta che un utente usa il programma e inserisce le credenziali di Dropbox e Drivefarm, queste sono memorizzate in un database locale. Lo scopo è di velocizzare una seconda operzione di trasferimento saltando la procedura di login dei due sistemi.

Qualora l'utente però volesse cambiare uno dei due, o entrambi i profili dovrà cliccare sul button "impostazioni". Potrà così accedere a una schermata (Figura 4.11) in cui sono visualizzate le due mail con le quali ha eseguito l'accesso e, a suo piacimento, rimuovere quella desiderata.

3.2.2 Sorgente

Il codice sorgente di Data Importer comprende packages e moduli in linguaggio python. Per quanto riguarda la parte visibile del programma (finestre, button, progressbar...) è stata utilizzata la libreria Qt.

zero12.dataimporter.application.py

Il modulo application.py, che risiede nel package dataimporter, a sua volta dentro zero12, è il modulo principale. Da qui viene gestito tutto il programma. È composto da una classe chiata "MainWindow", un derivato di "QMainWindow", del package "QtGui". Questa classe dirige e coordina le varie finestre dell'applicazione. Eseguendo application.py, di fatto, viene creata un'istanza di "MainWindow()".

All'interno del costruttore, "MainWindow" inizializza le variabili d'istanza, setta il titolo e immagine del programma, stabilisce le dimensioni della finestra e legge il file "style.qss", foglio di stile per il layout dell'applicazione.

Welcome

Ultima operazione del costruttore è chiamare la funzione "welcome_function()"; tale funzione crea un'istanza della classe "Welcome", che deriva da "QtGui.QWidget". In questa classe risiede il codice della prima schermata del programma.

Altra azione che compie è controllare che un precedente trasferimento con "Data Importer" non sia andato a buon fine. Se verifica questo, lancia una finestra di dialogo per rendere l'utente a conoscenza del problema. Il codice della finestra di dialogo è presente anch'esso nel modulo "application.py". Visualizzato il messaggio l'utente può scegliere di riprendere il trasferimento da dove si era interrotto e quindi verranno settati tutti i parametri in modo adeguato.

DropboxLogin

Nel caso in cui non vi siano state trovate anomalie, l'utente può premere "inizia" e far partire perciò la funzione "dropbox_login_function()". La funzione accede al database locale del programma che consiste in un file di tipo testo. Si tratta di un database non relazionale dove ogni dato è strutturato in documenti. Particolarità dei database non relazionali è che un documento può contenere altri documenti al suo interno. In questo caso, la funzione va in cerca di un dato con attributo "dropbox". L'attributo contiene il token di accesso alle chiamate per il sistema Dropbox.

Nel caso in cui non vi sia il database, oppure l'attributo "dropbox" non abbia valore, verrà istanziato un thread e visualizzata una finestra di dialogo. Nella finestra, appare all'utente il sito di accesso a Dropbox, dove, una volta inserite le credenziali e dato il consenso, sarà recuperato il token di accesso. Il thread, a questo punto, crea il database, se serve, e inserisce il token nell'attributo "dropbox".

GetDropboxTree

Recuperato o letto il token dal database, viene invocata la finzione "get_dropbox_tree_function()" con la successiva creazione di un'istanza di "GetDropTree". La classe appena presentata

contiene il codice per visualizzare la schermata di attesa mentre vengono caricati i file presenti in Dropbox in una struttura ad albero.

Di fatto, mentre l'utente visualizza una progressbar che scorre, la funzione "get_dropbox _tree_function()" invoca un thread. Scopo di questo thread è creare un'istanza della classe "MakeTree", appartenente al modulo "zero12.dataimporter.dropboxtree.py". La classe possiede un metodo ricorsivo che visita in preorder tutti i dati dell'utente nel suo profilo di Dropbox. Durante la visita i dati sono inseriti in un'altra classe, derivata da "QtCore.QAbstractItemModel". Un riferimento a un'istanza di quest'ultima classe viene collegato a una variabile nella "MainWindow".

In questo modo sarà possibile, una volta completato il metodo ricorsivo, avere un riferimento ad una struttura ad albero contenente tutti file e cartelle del profilo Dropbox dell'utente. Terminato il thread si passa alla funzione "dropbox_tree_function()".

DropboxTree

La classe "DropboxTree" viene istanziata ora. Al suo interno, nel layout della schermata, è possibile visualizzare il contenuto dell'albero grazie al riferimento acquisito con la funzione precedente. Per ogni dato presente è visualizzato il nome, un'icona che contraddistingue file e cartelle e una checkbox per dare la possibilità all'utende di essere selezionato.

DrivefarmLogin

Al click del button "avanti" viene invocata la funzione "drivefarm_login_function()" che, come prima operazione, controlla il numero di dati selezionati precedentemente. Un utente deve aver selezionato almeno un file da trasferire, se ciò non fosse successo parte nuovamente la funzione "dropbox_tree_function()", seguita da una finestra di dialogo il cui codice è scritto anch'esso nel modulo "application.py".

Quando, invece, il numero dei dati selezionati è maggiore di zero, "drivefarm_login_function()" continua la sua esecuzione eseguendo un'istanza del thread "GrowDropList". Questi ha lo scopo di inserire in un oggetto di tipo lista i file e le cartelle selezionati per facilitare poi il trasferimento.

Altra operazione che viene eseguita dopo il lancio del thread è la verifica nel database non relazionale se vi compare un documento con attributo "drivefarm", il cui valore corrisponde al token di accesso a Drivefarm. Trovato il token viene chiamata la funzione "get_drivefarm_tree_function()".

In caso contrario si procede istanziando la classe "DrivefarmLogin", che permetterà all'utente di inserire la mail e password del proprio profilo Drivefarm. Grazie a questi dati verrà ricavato il token e chiamata "get_drivefarm_tree_function()".

GetDrivefarmTree

In questa funzione si ripercorre il procedimento visto per "get_dropbox_tree_function()". L'istanza di GetDrivefarmTree genera una finestra di attesa per l'utente mentre viene chiamato un thread il quale visita il profile Drivefarm dell'utente e memorizza i dati in un'altra classe derivata da "QtCore.QAbstractItemModel", questa volta nel modu-

lo “zero12.dataimporter.drivefarmtree.py”. In questo modo viene memorizzato anche il riferimento alla struttura ad albero di Drivefarm.

DrivefarmTree

Al termine del processo la schermata d’attesa lascia il posto alla visualizzazione delll’albero, e per far questo viene creata un’istanza della classe “DrivefarmTree” dalla funzione “drivefarm_tree_function()”. Anche qui, come in precedenza, per ogni dato presente è visualizzato il nome, un’icona e una checkbox nelle cartelle. L’utente viene quindi invitato a selezionare la cartella dove desidera importare i file selezionati. Per questo motivo, in quest’albero, solo le cartelle hanno una checkbox.

GetSize

Il button “avanti” è collegato alla funzione “get_size_function()”. La funzione, come prima azione, controlla che sia stata selezionata solo una cartella. Se così non fosse l’utente verrebbe avvertito con una finestra di dialogo e si ritornerebbe alla schermata precedente.

Altrimenti viene istanziato un thread che recupera la path selezionata, memorizzandola anche nel database.

Altra operazione compiuta, dalla quale la funzione prende il nome, è istanziare la classe “GetSize” e con essa lanciare il thread omonimo. La classe permette all’utente di visualizzare una schermata con progressbar, allo scopo di segnalare all’utente che si stanno svolgendo delle operazioni. Il thread, infatti, esegue la funzione “iterSizeDb” che copia i dati di Dropbox selezionati nel database e ne calcola la dimensione totale. Successivamente viene calcolato anche lo spazio libero presente nel disco fisso e nel profilo Drivefarm per garantire l’effettiva migrazine dei dati.

ViewSize

Il risultato di questi calcoli è visualizzato grazie alla funzione “view_size()”, chiamata al termine del thread. La funzione, infatti, crea un’istanza della classe “ViewSize” che informa l’utente se il trasferimento dei dati selezionati da Dropbox nel profilo Drivefarm è possibile. Al click sul button “avanti” si procede col evocare la funzione “transfer_function()”.

Transfer

“transfer_function()” ha l’unico compito di creare un’istanza della classe “Transfer”. È la classe stessa che ha al suo interno il codice per gestire il trasferimento. All’inizio viene creata una progressbar e dato come indice massimo la dimensione dei dati da scaricare. Ad essa è associata la funzione “timerEvent()”, con il compito di visualizzare l’avanzamento della barra mano a mano che i dati vengono trasferiti. Tutto ciò però non ha inizio se l’utente non preme il button “inizia trasferimento” al quale è collegata un’altra funzione, “doAction()”.

L’ultima funzione nominata, quando viene chiamata, esegue un controllo sulla progressbar. Se è in stato di “pausa” istanzia due thread: “drop” e “drive”. “drop” ha il

compito di scaricare in una cartella temporanea nel disco fisso i dati di Dropbox, seguendo la lista creata precedentemente dalla funzione "drivefarm_login_function()". Allo stesso tempo, per ogni file scaricato, aggiorna una lista chiamata "commonList".

Comtemporaneamente il thread "drive" legge quanto scritto nella "commonList". In questo modo è sicuro che i dati scritti lì sono presenti anche nella cartella temporanea, li recupera e li trasferisce in Drivefarm. Oltre a questo aggiorna il database locale settando a True l'attributo "tranfer-state" del file trasferito.

Sfruttando questo processo lo scaricamento e il trasferimento avvengono in simultanea e non sequenzialmente, accelerando il tutto.

Se in un momento qualsiasi, l'utente cliccasse nuovamente sul button "inizia trasferimento", che all'avvio avrà preso il nome di "stop", viene nuovamente chiamata la funzione "doAction()". Ora, però, lo stato della progressbar non è più in "pausa". L'azione eseguita in questo caso sarà bloccare i due thread e chiedere all'utente, attraverso una finestra di dialogo, l'azione desiderata per procedere. Azioni disponibili sono: pausa, skip e annulla trasferimento. Cliccando su pausa il button riprenderà la scritta "inizia trasferimento" e sarà possibile riprendere il trasferimento dei file da dove era stato interrotto. Alla scelta di skip verranno eseguite le stesse procedure con la differenza che il file attualmente in trasferimento verrà saltato. L'ultima opzione invece farà partire un thread, il cui codice risiede nel modulo "zero12.dataimporter.annul.py", adibito alla rimozione dei dati già scaricati in Drivefarm.

Un'evenienza che può verificarsi è il ritrovamento di un file o cartella con il nome uguale a quello in trasferimento. Ne segue che il thread "drive" viene messo in wait ed anche in questo caso compare una finestra di dialogo per chiedere all'utente l'azione desiderata tra: rinomina, sovrascrivi e skip. Il codice da eseguire una volta individuata la scelta è presente all'interno del thread stesso.

Completato il trasferimento il database locale viene svuotato e la cartella temporanea eliminata. Inoltre nella schermata principale diventa cliccabile un button con scritto "avanti", grazie al quale si ritorna alla schermata iniziale del programma.

Se durante il trasferimento si verificasse la chiusura del programma, data da un'azione volontaria o involontaria dell'utente, al nuovo riavvio "Data Importer" è in grado di completare il trasferimento interrotto. Se infatti terminasse la batteria del portatile, o l'utente arrestasse il sistema del computer, nel database locale rimarrebbero salvati i dati non ancora correttamente trasferiti. Il controllo del database è fatto al momento dell'avvio dalla classe"Welcome".

RemoveAccount

La classe "RemoveAccount" è istanziata dalla funzione "remove_account_function()" alla pressione del button "impostazioni" presente in ogni sezione del programma. L'effeto è la visualizzazione di un schermata in cui appaiono le mail usate per l'accesso a Dropbox e Drivefarm e due button con la scritta "rimuovi" sotto ad esse. Tali button sono collegati rispettivamente alle funzioni "del_drop_function()" e "del_drive_function()" che cancellano dal database locale, la prima, il token di accesso a Dropbox, la seconda quello a Drivefarm. In questo modo sarà possibile inserire un account diverso da quello indicato precedentemente.

Capitolo 4

Conclusioni

Le due applicazioni sono state sviluppate nei termini temporali stabiliti. Per ogniuna di esse è stato impiegato un mese per la progettazione e realizzazione.

Gli obiettivi base di entrambi i moduli sono stati raggiunti, ora i programmi sono attivi e funzionanti.

Gmail Plugin è stato presentato in occasione del Web Summit 2013 a Dublino, evento di importanza europea in termini di tecnologia e innovazione. A questo Summit, infatti, hanno fatto il loro intervento 350 speaker di fama internazionale ed è stato allestito il settore Alpha, dove quasi mille startupper europee hanno potuto esporre il proprio lavoro a media, pubblico e investitori.

Capitolo 5

Bibliografia e Sitografia

- www.zero12.com
- www.drivefarm.com
- developers.google.com/gmail
- www.dropbox.com/developers
- www.mongodb.org
- 2013.websummit.net