Praxismodul: Duplikateliminator

1.Zielbestimmung

Es ist eine Softwarelösung zu entwickeln, die einen vorgegebenen Datenbestand nach bestimmten Kriterien durchsucht, die einzelnen Dateiattribute in eine Datenbank schreibt und soweit möglich selbstständig bei Verletzung dieser Kriterien vordefinierte Aktionen ausführt. Solche Kriterien können beispielsweise doppelte Dateien sein, oder Dateien die seit langem nicht mehr benutzt wurden.

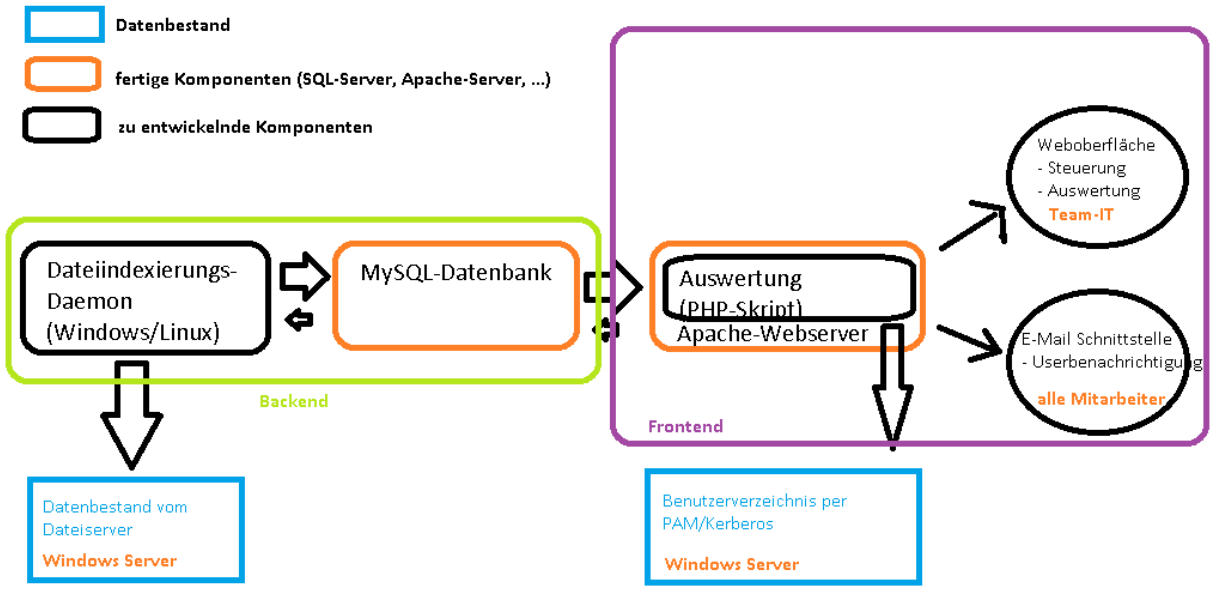
Mögliche Aktionen sollen beispielsweise das Benachrichtigen des Eigentümers der Datei (E-Mail, SMS, etc.), dass Entfernen der Duplikate (Löschen der Duplikate, ersetzen der Duplikate durch Links, etc.) sein oder im Zweifelsfall ein Eskalationskonzept (Bsp: weigert sich der Verursacher oder fällt wiederholt auf, Gespräch mit Abteilungsleiter-IT, dann Geschäftsführer.). Zur administrativen Verwaltung und Wartung des Systems ist eine Web-Oberfläche zu entwickeln, die auch nützliche statistische Auswertungen zur Verfügung stellen soll.

Ziel ist also die Vermeidung von unerwünschter Redundanz und damit verbunden die Entlastung der Speicher- und Backupsysteme.

2.Produkteinsatz

Die Lösung soll möglichst effizient vermeiden das sich alte oder doppelte Dateien auf den Dateiservern ansammeln. Weiterhin soll die Lösung weitgehend selbstständig arbeiten um die Mitarbeiter in der It-Abteilung zu entlasten und nur im Ausnahmefall eine administrative Interaktion erfordern. Aktuelle Untersuchungen im Unternehmen haben ergeben das mehr als 50GB an Speicherplatz nur durch Duplikate belegt werden.

3.Produktübersicht



Indexierungsdienst mit MySQL-Anbindung:

- Folgende Optionen sind zu prüfen:
    - fdupes (GPL-Lizenz) um MySQL-Anbindung erweitern
        - siehe dazu MySQL Connector/C(++): http://forge.mysql.com/wiki/Connector_C%2B%2B
    - eigenen Indexierungsdienst mit MySQL-Anbindung entwickeln

Verlauf: Punkt 1 wurde nach hinreichender Prüfung verworfen, da fdupes Linux als Host-Betriebssystem vorraussetzt, die zu untersuchenden Verzeichnisse aber ActiveDirectory-Ressourcen einer Windowsdomäne sind. Es bieten sich 2 Möglichkeiten diese Ressourcen unter Linux nutzbar zu machen:

- Einbinden der Windowsfreigaben per Cifs-Mount im lokalen Kontext des Linuxnutzers
    - Nachteil: sämtliche Dateiattribute wie zB der ursprüngliche Dateieigentümer gehen verloren da sie beim mounten durch den lokalen Linuxbenutzer ersestzt werden, in dessen Kontext das mounten stattfindet
- Einbinden des Linuxhosts direkt in die ActiveDirectory-Domäne
    - Wenn überhaupt nur sehr umständlich und mit viel Improvisation möglich, daher für den Produktiveinsatz für nicht praktikabel befunden

Somit scheidet Punkt 1 aus und es wird eine komplette Neuimplementierung umgesetzt. Da die komplette IT-Landschaft der Firma über eine ActiveDirectory-Domäne abgebildet wird und als Betriebssysteme fast ausschließlich Microsoft-Produkte zum Einsatz kommen bietet sich eine Implementierung des Indexierungsdienstes in C# an.

Funktionsumfang des Indexierungsdienstes:

- Rekursives Durchlaufen vorgegebener Verzeichnisse
- Erfassung sämtlicher Dateiattribute der vorgfundenen Dateien sowie Bildung von Prüfsummen um später Duplikate erkennen zu können
- Berechung der Prüfsummen unter Zuhilfenahme der Grafikkarte (Cuda, OpenCL, etc)
- Ausgabe dieser gesammelten Metadaten über eine SQL-Schnittstelle, zB für MySQL oder MSSQL-Server
- Fortschritt der Implementierung in C#: lauffähiger Prototyp ohne die Funktionen: SQL-Schnittstelle, Berechnen der Prüfsumme mit Hilfe der Grafikkarte
- Projekt ist gehostet auf github.com: https://github.com/mkirbst/cworker

Quellcodelisting: Prototyp Dateiindexierungsdienst

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
using System.Security.Principal;

/** Quellenverzeichnis:
 *  walkFolders:    http://dotnet-snippets.de/dns/rekursiver-verzeichnislauf-SID462.aspx
 *  md5:            http://dotnet-snippets.de/dns/md5-hash-von-dateien-ermitteln-SID66.aspx
 *
 */

namespace worker  {
    class Program    {
        static void Main(string[] args)   {
            //walkFolders("q:\\");
            walkFolders("C:\\Users\\kirbst\\Desktop\\buecher");

        }

        private static void walkFolders(string Directory)  {
            walkFolders(new DirectoryInfo(Directory));
        }

        private static void walkFolders(DirectoryInfo di)  {
            try  {
                // Alle Verzeichnisse rekursiv durchlaufen
                foreach (DirectoryInfo subdir in di.GetDirectories())  {
                    walkFolders(subdir);
                }

                // Alle Dateien durchlaufen
                foreach (FileInfo fi in di.GetFiles())
                {
                      Console.Write(hrs(fi.Length) + " " +
                      File.GetAccessControl(@fi.FullName).GetOwner(typeof(NTAccount)) + " " +
                      fi.FullName + "  SHA512: [" + Datei2SHA(fi.FullName) + "]" + "  MD5: ["
                      + Datei2MD5(fi.FullName) + "]" + "\r");
                }
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
            }
        }

        /**md5
         */
        public static string Datei2MD5(string Dateipfad)
        {
            //Datei einlesen
            System.IO.FileStream FileCheck = System.IO.File.OpenRead(Dateipfad);
            // MD5-Hash aus dem Byte-Array berechnen
            System.Security.Cryptography.MD5 md5 = new
System.Security.Cryptography.MD5CryptoServiceProvider();
            byte[] md5Hash = md5.ComputeHash(FileCheck);
            FileCheck.Close();

            //in string wandeln
            string Berechnet = BitConverter.ToString(md5Hash).Replace("-", "").ToLower();
            return Berechnet;

        }

        /**sha512
         */
        public static string Datei2SHA(string Dateipfad)
        {
            //Datei einlesen
            System.IO.FileStream FileCheck = System.IO.File.OpenRead(Dateipfad);
            System.Security.Cryptography.SHA512 sha512
                                = new System.Security.Cryptography.SHA512CryptoServiceProvider();
            byte[] sha512Hash = sha512.ComputeHash(FileCheck);
            FileCheck.Close();
```

```
            //in string wandeln
            string Berechnet = BitConverter.ToString(sha512Hash).Replace("-", "").ToLower();
            return Berechnet;

        }

        //humanreadable filesize
        public static string hrs(long filebytes)  {
            string hrsize = "";
            int n = 0, nachkomma = 0;

            while (filebytes > 1024)
            {
                nachkomma = (int)(filebytes % 1024);
                filebytes /= 1024;
                n++;
            }
            hrsize = filebytes.ToString();
            if (nachkomma > 100)
            {
                nachkomma = nachkomma / 100;
                hrsize += "," + nachkomma.ToString();
            }
            switch (n)
            {
                case 1: hrsize += "KB"; break;  // " KiloByte"; break;
                case 2: hrsize += "MB"; break;  // " MegaByte"; break;
                case 3: hrsize += "GB"; break;  // " GigaByte"; break;
                case 4: hrsize += "TB"; break;  //" TerraByte"; break;
                case 5: hrsize += "PB"; break;  //" PetaByte"; break;
                case 6: hrsize += "EB"; break;  //" ExaByte"; break; /*long: file.attrs can be
max. 9 exabyte*/
                default: hrsize +="B "; break;  //" Byte"; break;
            }
            return hrsize;
        }
    }
}
```

Ausgabe beispielhaft:

```
2,6MB BAMBERG\kirbst
C:\Users\kirbst\Desktop\buecher\betriebssysteme\linux\Linux
_Hochverfuegbarkeit.pdf  SHA512:
[66bff6b3d5bc5706fba5d981ed006d5dc2507d05bc145f
5db12f47716092e369bc20f1070e51732c8bb3d6ea26f1ff1884a3a1581389037233badfec1
c77db]  MD5: [049b8cea80c58bb33686ad11e5439690]
1B  BAMBERG\kirbst
C:\Users\kirbst\Desktop\buecher\betriebssysteme\linux\TEST.tx
t  SHA512:
[1f40fc92da241694750979ee6cf582f2d5d7d28e18335de05abc54d0560e0f530286
0c652bf08d560252aa5e74210546f369fbbbce8c12cfc7957b2652fe9a75]  MD5:
[0cc175b9c0f1b6a831c399e269772661]
4,9MB BAMBERG\kirbst
C:\Users\kirbst\Desktop\buecher\betriebssysteme\linux\Wolfi
nger Gulbins Hammer - Linux Systemadministration - Grundlagen, Anwendungen
und K
onzepte.pdf  SHA512:
[ca354af054ac1fe07edee5265fbfcc5c63679c19f3c244722997b470ee
fae04531437471f654a9ed67315139195df5be7270dba3e1fba1ed5089916b1be16258]
MD5: [62aa61a32ba48353466b6a64c4f2c3cf]
```

Die Ausgabe ist beispielhaft so schematisiert:
Dateigröße Eigentümer Dateiname mit Pfad Prüfsumme SHA-512 Prüfsumme MD5

MySQL-Datenbank:

Datensatzattribute(vorläufig):

Datenattribut	Datentyp(MySQL)	Einheit	Beschreibung
timestamp	Zeitpunkt (DATETIME)	Sek. Seit 1.1.1970 (Unix-Zeitformat)	Zeitpunkt der erstellung des Datensatzes
checksum	String	MD5/SHA Prüfsumme	
files	Integer (UNSIGNED INT)		Anzahl Dateiexemplare
Path	String	Pfad1\Name1;[Pfad2\Name2;…;]	Pfade (beinhalten auch Dateiname selbst) zu den Dateien
Size	Long Int (UNSIGNED BIGINT)	Byte	Größe der Dateien
Owner	String	Eigentuemer1;[Eigentuemer2;…;]	Eigetümeruser der Datei
Group	String	Gruppe1;[Gruppe2;…;]	Eigentümergruppe der Datei
atime	Zeitpunkt (DATETIME)	Sek. Seit 1.1.1970 (Unix-Zeitformat)	Zeitpunkt des letzten Lesezugriffs
mtime	Zeitpunkt (DATETIME)	Sek. Seit 1.1.1970 (Unix-Zeitformat)	Zeitpunkt des letzten Schreibzugriffs

Primärschlüssel:

- Format: [Pruefsumme][Path]
- Begründung: In der Praxis soll der Indexierungsdienst periodisch laufen (zB wöchentlich).In dieser Zeit können die Dateien modifiziert worden sein und weist dann auch eine andere Prüfsumme auf. Über den Zeitstempel kann man in der Datenbank aber dann vor der Neuindexierung alle alten Datensätze verwerfen.

Bsp: Erzeuge MySQL-Tabelle:

```
CREATE TABLE duplicates (
        tstamp  TIMESTAMP,
        csum    VARCHAR(32) ASCII
        files   UNISGNED INT,
        path    VARCHAR(512) UNICODE,
        size    UNSIGNED BIGINT,
        owner   VARCHAR(64) UNICODE,
        group   VARCHAR(64) UNICODE,
        atime   TIMESTAMP,
        mtime TIMESTAMP
);
```

Auswertungsmodul:

-TODO-