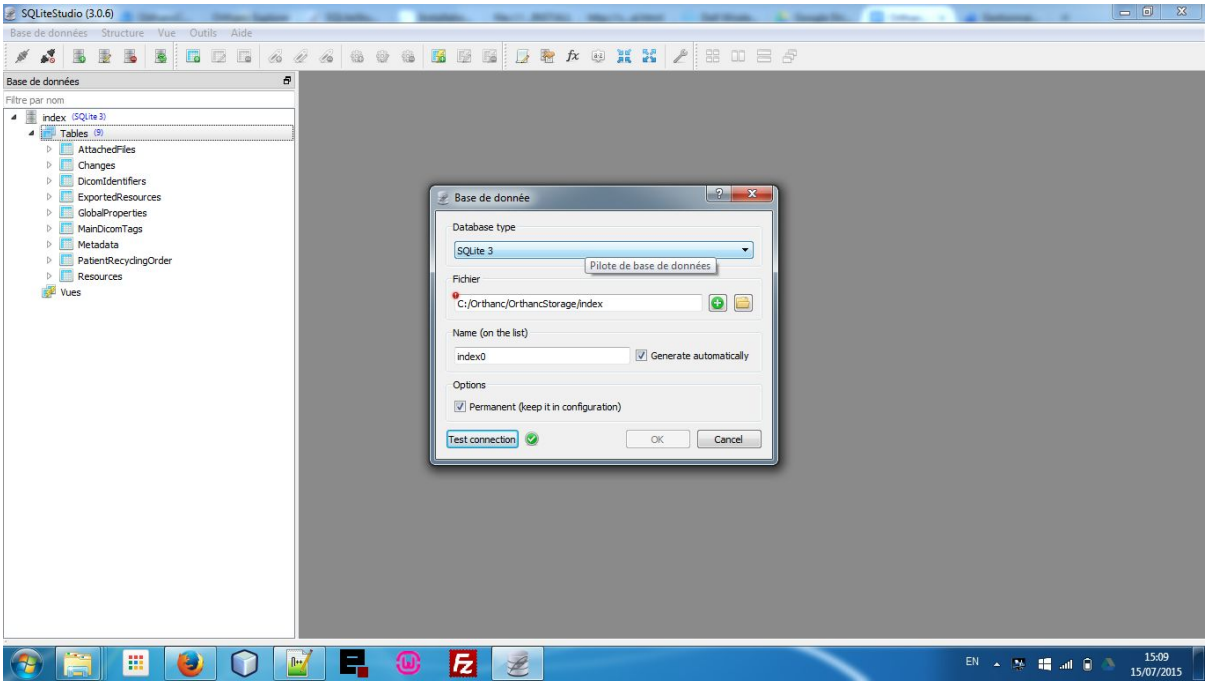
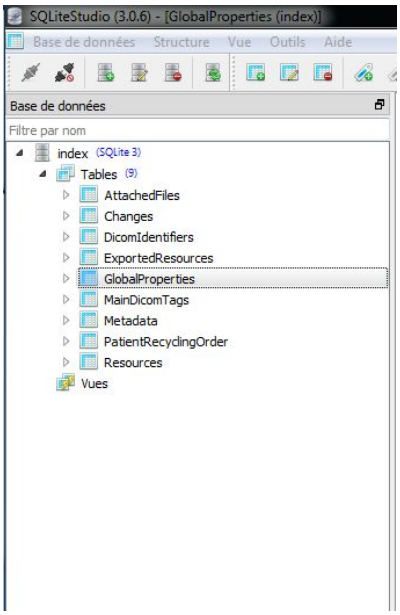
Dans cette partie je vais analyser le code C++ d'Orthanc ainsi que sa base SQL :

Tout d'abord je suis allé voir le code SQL d'Orthanc (C:\Users\syndrome\Downloads\TT DICOM\Code source Orthanc\jodogne\Releases\Orthanc-0.9.1\OrthancServer\PrepareDatabase.sql+Upgrade3to4.sql+Upgrade4to5.sql).
Pour étudier la structure de données d'Orthanc qui utilise SQLite, on peut utilise SQLiteStudio pour gérer ces données. (Lien : http://sqlitestudio.pl/).
Une fois tout installé, on lance SQLiteStudio, puis dans l'onglet "Base de données", on ajoute le fichier voulu (celui qui contient la base de données sqlite d'Orthanc) : C:\Orthanc\OrthancStorage\index
(NB : ce fichier n'a pas d'extension)



Notre base de données devrait donc s'afficher:



C'est ainsi qu'en ajoutant un fichier DICOM pour vétérinaire, j'ai pu remarquer que seuls les tags pour des patients humains étaient pris en compte. J'ai alors remarqué que les tags étaient identifiés grâce à son groupe et son élément :

```
CREATE TABLE MainDicomTags(
    id INTEGER REFERENCES Resources(internalId) ON DELETE CASCADE,
    tagGroup INTEGER,
    tagElement INTEGER,
    value TEXT,
    PRIMARY KEY(id, tagGroup, tagElement)
    );
```

Certains tags sont aussi prit à part afin de les insérer dans la table DicomIdentifiers qui représente les identifiants du patient.

```
INSERT INTO DicomIdentifiers SELECT * FROM MainDicomTags
      WHERE ((tagGroup = 16 AND tagElement = 32) OR  -- PatientID (0x0010, 0x0020)
             (tagGroup = 32 AND tagElement = 13) OR  -- StudyInstanceUID (0x0020, 0x000d)
             (tagGroup = 8  AND tagElement = 80) OR  -- AccessionNumber (0x0008, 0x0050)
             (tagGroup = 32 AND tagElement = 14) OR  -- SeriesInstanceUID (0x0020, 0x000e)
             (tagGroup = 8  AND tagElement = 24));   -- SOPInstanceUID (0x0008, 0x0018)

DELETE FROM MainDicomTags
      WHERE ((tagGroup = 16 AND tagElement = 32) OR  -- PatientID (0x0010, 0x0020)
             (tagGroup = 32 AND tagElement = 13) OR  -- StudyInstanceUID (0x0020, 0x000d)
             (tagGroup = 8  AND tagElement = 80) OR  -- AccessionNumber (0x0008, 0x0050)
             (tagGroup = 32 AND tagElement = 14) OR  -- SeriesInstanceUID (0x0020, 0x000e)
             (tagGroup = 8  AND tagElement = 24));   -- SOPInstanceUID (0x0008, 0x0018)
```

C'est pour cela que j'ai décidé d'aller observer le code C++. Voici les fichiers importants que j'ai pu remarquer :

- C:\Users\syndrome\Downloads\TT DICOM\Code source Orthanc\jodogne\Releases\Orthanc-0.9.1\Core\Dicomtag.cpp+Dicomtag.h

On peut observer dans ces fichiers tous les tags qui ont été définies dans différents modules de l'étude de cas (Patient, Study, Series, Instance)

- C:\Users\syndrome\Downloads\TT DICOM\Code source Orthanc\jodogne\Releases\Orthanc-0.9.1\OrthancServer\DatabaseWrapper.cpp

C'est ici que j'ai pu observer les échanges SQLite qui se faisaient au niveau des tags :

```
481  void DatabaseWrapper::SetMainDicomTag(int64_t id,
482                                        const DicomTag& tag,
483                                        const std::string& value)
484  {
485    if (tag.IsIdentifier())
486    {
487      SQLite::Statement s(db_, SQLITE_FROM_HERE, "INSERT INTO DicomIdentifiers VALUES(?, ?, ?, ?)");
488      SetMainDicomTagsInternal(s, id, tag, value);
489    }
490    else
491    {
492      SQLite::Statement s(db_, SQLITE_FROM_HERE, "INSERT INTO MainDicomTags VALUES(?, ?, ?, ?)");
493      SetMainDicomTagsInternal(s, id, tag, value);
494    }
495  }
496
497  void DatabaseWrapper::GetMainDicomTags(DicomMap& map,
498                                         int64_t id)
499  {
500    map.Clear();
501
502    SQLite::Statement s(db_, SQLITE_FROM_HERE, "SELECT * FROM MainDicomTags WHERE id=?");
503    s.BindInt64(0, id);
504    while (s.Step())
505    {
506      map.SetValue(s.ColumnInt(1),
507                   s.ColumnInt(2),
508                   s.ColumnString(3));
509    }
510
511    SQLite::Statement s2(db_, SQLITE_FROM_HERE, "SELECT * FROM DicomIdentifiers WHERE id=?");
512    s2.BindInt64(0, id);
513    while (s2.Step())
514    {
515      map.SetValue(s2.ColumnInt(1),
516                   s2.ColumnInt(2),
517                   s2.ColumnString(3));
518    }
519  }
520
```

On peut voir ici qu'il n'y a toujours pas la réponse à notre question : " Pourquoi est-ce que les tags spécial aux animaux ne sont pas pris en compte ? (Patient breed, patient species …) "

C'est pour cela que nous avons décidé d'aller analyser Orthanc sur une autre couche, la couche HTML à l'aide de FireFox (Allez partie : **Analyse de GET de l'upload et la lecture d'un fichier Dicom**)