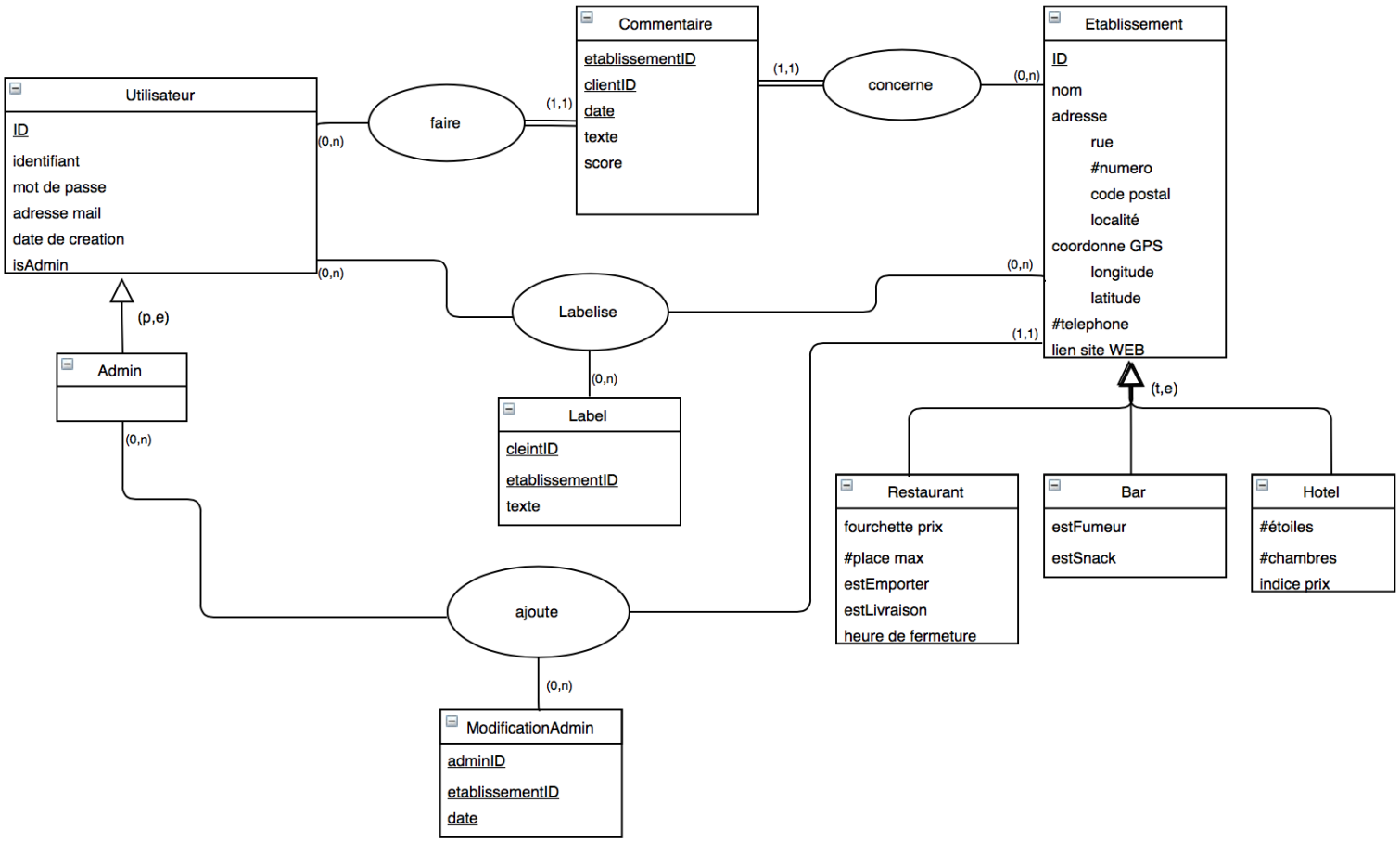
Rapport : Projet - Base de donnée

Maximilien ROMAIN (000411776) - George RUSU (000407965)

Le 8 mai 2016

Table des matières

1 Diagramme



1.1 Contraintes :

— Un client peut être un admin ou bien un utilisateur.

— Un client ne peut pas commenter un même établissement une même date.

— La date d'enregistrement d'un utilisateur doit être inférieure a la date de chaque commentaire qu'il fait.

— Un client peut faire soit un commentaire soit apposer un label(tag) qu'il crée ou qu'il invente.

— Un commentaire doit contenir l'ID du client, l'ID de l'établissement ainsi que le texte et le score.

— Un label doit contenir l'ID du client, l'ID de l'établissement ainsi que le label(tag).

— Un client ne peut pas labeliser plusieurs fois un même établissement avec un même label.

2 Model Relationnel

— Etablissement(ID, Nom, Adresse, AdRue, AdNuméro, AdCodePostal, AdLocalité, Coordonnée, CoodLongitude, CoordLatitude, NumTelephone, *SiteWeb*)

— Restaurant(ID, FourchettePrix, NbrPlace, *Emporter*, *Livraison*, Fermeture)
ID référence Etablissement.ID

— Bar(ID, *Fumer*, *Snack*)
ID référence Etablissement.ID

— Hotel(ID, NbrEtoiles, NbrChambres, IndicePrix)
ID référence Etablissement.ID

— Utilisateur(ID, Identifiant, AdresseMail, MotDePasse, DateCréation)

— Admin(ID)
ID référence Utilisateur.ID

— ModificationAdmin(AdminID, EtablissementID, Date)
AdminID référence Utilisateur.ID ; EtablissementID référence Etablissement.ID

— Commentaire(Score, Texte, Date, EtablissementID, UtilisateurID)
EtablissementID référence Etablissement.ID ; UtilisateurID référence Client.ID

— Label(EtablissementID, UtilisateurID, Tag)
EtablissementID référence Etablissement.ID ; UtilisateurID référence Utilisateur.ID ;

3 Choix d'implémentation

Pour expliquer notre schéma UML plus haut, nous avons choisi l'implémentation suivante : lorsqu'on voudra connaître les informations d'un établissement en particulière (commentaires, tags , auteur du commentaire, auteur du tag, etc) il nous suffira de retrouver son id correspondant et de regarder avec cet id dans les tables Commentaire ,Label et ModificationAdmin. Idem, pour un utilisateur, si on souhaite savoir ce qu'il a commenté. Nous pouvons donc dire que nos deux tables de bases sont Commentaire et Label.

En ce qui concerne le distinction entre utilisateur et admin, nous avons une colonne, isAdmin qui grâce à un booléen ou bien à un int (0 ou 1) nous permet de savoir si le client est admin ou bien un simple utilisateur. L'héritage présent dans le schéma permet donc de voir ce qu'un admin peut faire en plus qu'un client standard, c'est à dire creer un établissement.

Les langages utilisé sont : le "html" et "css3" ainsi que le "php" et pour ce qui concerne le sql nous avons choisi le "mysql".

4 Installation

Pour installer notre application, il est d'abord nécéssaire de mettre en place la base de données qui contiendra toutes les informations qui seront utilisées. Pour ce faire le premier fichier à exécuter une fois que la connexion à un serveur mysql est faite, est "Create-DB.php", en vérifiant que le fichier header_connect.php se connectera bien. Une fois cela fait, la base de donnée Eureka à été crée ainsi que toutes ses tables. Afin de ne pas commencer avec une application vide, nous allons parser des fichiers XML qui vont remplir notre base de donné en respectant les contrainte cité plus haut(date des utilisateurs). L'application est maintenant bien installé et il est possible désormais de l'utiliser correctement. Notre application est un site, il est donc maintenant possible de lancer la page "index.php" et nous voila dans notre application.

5 Script SQL DDL

Voici le fichier "Create-DB.php" comme demandé :

```
<?php
        include ("header_connect.php");
//Creation de Eureka
        echo "Creating Database status: ";
        try {
        $conn = new PDO("mysql:host=$servername", $username, $password);
        $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        $sql = "CREATE DATABASE Eureka";
        $conn->exec($sql);
        echo "Database created successfully<br>";

                }
        catch(PDOException $e) {
        echo "Error: " . $e->getMessage();
        echo "<br>";
        }
        $sql = "USE Eureka";
    //selecting DB
    $conn->exec($sql);
        echo "Creating table Utilisateur status: ";
        try {
        $sql = "CREATE TABLE Utilisateur (ID INT UNSIGNED NOT NULL AUTO_INCREMENT,
        identifiant VARCHAR(30) NOT NULL UNIQUE, mot_de_passe VARCHAR(30) NOT NULL,
        email VARCHAR(50) NOT NULL, dateCreation DATE NOT NULL, isAdmin TINYINT(1) NOT NULL DEFAULT 0,
        PRIMARY KEY (id,identifiant))ENGINE=InnoDB";
        $conn->exec($sql);
        echo "Table created successfully<br>";

                }
        catch(PDOException $e) {
        echo "Error: " . $e->getMessage();
        echo "<br>";
        }

        echo "Creating table Etablissement status: ";
        try {

        $sql = "CREATE TABLE Etablissement (ID INT UNSIGNED NOT NULL AUTO_INCREMENT,
                nom VARCHAR(50) NOT NULL UNIQUE, rue VARCHAR(50) NOT NULL, numero INT UNSIGNED NOT NULL,
                codePostal INT UNSIGNED NOT NULL, localite VARCHAR(30) NOT NULL, longitude FLOAT NOT NULL,
                latitude FLOAT NOT NULL, telephone VARCHAR(15) NOT NULL, lienWeb VARCHAR(100),
                type VARCHAR(10) NOT NULL,PRIMARY KEY (ID,nom,type))ENGINE=InnoDB";
        $conn->exec($sql);
        echo "Table created successfully<br>";

                }
        catch(PDOException $e) {
        echo "Error: " . $e->getMessage();
        echo "<br>";
        }

        echo "Creating table Bar status: ";
        try {
        $sql = "CREATE TABLE Bar (ID INT UNSIGNED PRIMARY KEY NOT NULL, fumeur TINYINT(1) NOT NULL,
                petiteRestauration TINYINT(1) NOT NULL,FOREIGN KEY (ID) REFERENCES Etablissement(ID))
                        ENGINE=InnoDB";
        $conn->exec($sql);
        echo "Table created successfully<br>";

                }
        catch(PDOException $e) {
        echo "Error: " . $e->getMessage();
        echo "<br>";
        }
        echo "Creating table Hotel status: ";
        try {
        $sql = "CREATE TABLE Hotel (ID INT UNSIGNED PRIMARY KEY NOT NULL, prix FLOAT UNSIGNED NOT NULL,
                nbChambres INT UNSIGNED NOT NULL, nbEtoiles INT UNSIGNED NOT NULL,
                FOREIGN KEY (ID) REFERENCES Etablissement(ID))ENGINE=InnoDB";
        $conn->exec($sql);
        echo "Table created successfully<br>";

                }
        catch(PDOException $e) {
        echo "Error: " . $e->getMessage();
        echo "<br>";
        }
```

```
        echo "Creating table Restaurant status: ";
        try {
        $sql = "CREATE TABLE Restaurant (ID INT UNSIGNED PRIMARY KEY NOT NULL, prix FLOAT UNSIGNED NOT NULL,
                placesBanquet INT NOT NULL, emporter TINYINT(1) NOT NULL, livraison TINYINT(1) NOT NULL,
                fermeture VARCHAR(7) NOT NULL,FOREIGN KEY (ID) REFERENCES Etablissement(ID))ENGINE=InnoDB";
        $conn->exec($sql);
        echo "Table created successfully<br>";

                }
        catch(PDOException $e) {
        echo "Error: " . $e->getMessage();
        echo "<br>";
        }
        echo "Creating table ModificationAdmin status: ";
        try {
        $sql = "CREATE TABLE ModificationAdmin (etablissementID INT UNSIGNED NOT NULL,
                adminID INT UNSIGNED NOT NULL, dateCreation DATE NOT NULL,
                PRIMARY KEY (etablissementID, adminID, dateCreation),
                FOREIGN KEY (etablissementID) REFERENCES Etablissement(ID),
                FOREIGN KEY (adminID) REFERENCES Utilisateur(ID))
                        ENGINE=InnoDB";
        $conn->exec($sql);
        echo "Table created successfully<br>";

                }
        catch(PDOException $e) {
        echo "Error: " . $e->getMessage();
        echo "<br>";
        }
        echo "Creating table Label status: ";
        try {
        $sql = "CREATE TABLE Label (etablissementID INT UNSIGNED NOT NULL,
                clientID INT UNSIGNED NOT NULL, texte VARCHAR(50) NOT NULL,
                PRIMARY KEY (etablissementID, clientID, texte),
                FOREIGN KEY (etablissementID) REFERENCES Etablissement(ID),
                FOREIGN KEY (clientID) REFERENCES Utilisateur(ID))ENGINE=InnoDB";
        $conn->exec($sql);
        echo "Table created successfully<br>";

                }
        catch(PDOException $e) {
        echo "Error: " . $e->getMessage();
        echo "<br>";
        }
        //Creation de table Commentaire
        echo "Creating table Commentaire status: ";
        try {
        $sql = "CREATE TABLE Commentaire (etablissementID INT UNSIGNED NOT NULL,
                clientID INT UNSIGNED NOT NULL, dateCreation DATE NOT NULL,
                texte VARCHAR(1000) NOT NULL, score INT(1) NOT NULL,
                PRIMARY KEY (etablissementID, clientID, dateCreation),
                FOREIGN KEY (etablissementID) REFERENCES Etablissement(ID),
                FOREIGN KEY (clientID) REFERENCES Utilisateur(ID))ENGINE=InnoDB";
        $conn->exec($sql);
        echo "Table created successfully<br>";

                }
        catch(PDOException $e) {
        echo "Error: " . $e->getMessage();
        echo "<br>";
        }

?>
```

6 Requetes

6.1 En SQL

6.1.1 R1

```
SELECT identifiant FROM Utilisateur WHERE ID IN (SELECT clientID FROM
Commentaire WHERE etablissementID IN (SELECT etablissementID FROM
Commentaire WHERE clientID IN (SELECT ID FROM Utilisateur WHERE
identifiant="Brenda") AND score>=4) GROUP BY clientID HAVING
count(etablissementID)>=3)
```

6.1.2 R2

```
SELECT nom FROM Etablissement WHERE ID IN(SELECT DISTINCT
etablissementID FROM Commentaire WHERE clientID IN (SELECT
DISTINCT clientID FROM Commentaire WHERE etablissementID IN
(SELECT etablissementID FROM Commentaire WHERE clientID IN
(SELECT ID FROM Utilisateur WHERE identifiant="Brenda") AND
 score>=4) group by clientID having count(DISTINCT
 etablissementID)>=(SELECT count(etablissementID) FROM Commentaire
 WHERE clientID IN (SELECT ID FROM Utilisateur WHERE
   identifiant="Brenda")
 AND score>=4)))
```

6.1.3 R3

```
SELECT DISTINCT e.nom FROM Etablissement e WHERE NOT EXISTS
(SELECT * FROM Commentaire c WHERE c.etablissementID=e.ID) OR EXISTS
(SELECT * FROM Commentaire c WHERE c.etablissementID=e.ID GROUP BY e.ID
 HAVING count(*)<=1)
```

6.1.4 R4

```
SELECT identifiant FROM Utilisateur WHERE ID IN (SELECT m.adminID FROM
ModificationAdmin m WHERE not exists (SELECT * FROM Commentaire WHERE
  clientID=m.adminID AND etablissementID=m.etablissementID))
```

6.1.5 R5

```
SELECT nom FROM Etablissement e, Commentaire c WHERE
e.ID=c.etablissementID GROUP BY c.etablissementID HAVING count(*)>=3
 ORDER BY avg(c.score) DESC
```

6.1.6 R6

```
SELECT l.texte FROM Label l, Commentaire c WHERE
l.etablissementID=c.etablissementID GROUP BY texte HAVING
count(DISTINCT l.etablissementID)>=5 ORDER BY avg(c.score) DESC
```

6.2 Algèbre relationnelle

6.2.1 R1

$a \leftarrow \sigma_{identifiant="Brenda"}(Utilisateur)$
$b \leftarrow \sigma_{clientID\ IN(a) \wedge score>=4}(Commentaire)$
$c \leftarrow \sigma_{etablissementID\ IN(b) \wedge \#Etablissement>=3}(Commentaire)$
$d \leftarrow \sigma_{ID\ IN(c)}(Utilisateur)$

6.2.2 R2

$a \leftarrow \pi(\sigma_{Identifiant="Brenda"}(Utilisateur))$
$b \leftarrow a * Commentaire$
$c \leftarrow \sigma_{score>3}(b)$

$d \leftarrow \sigma_{score>3}(Commentaire)$
$e \leftarrow \pi_{etablissementID,clientID}(d - c)$
$f \leftarrow e \div c$
$g \leftarrow \pi_{Nom}(f \bowtie_{etablissementID=ID} Etablissement)$

6.2.3 R3

$a \leftarrow \pi_{ID}(Etablissement)$
$b \leftarrow \sigma_{etablissemntID\ NOT\ IN\ (a)\ OR\ \#clientID\ <=\ 1}(Commentaire)$

6.2.4 R4

$a \leftarrow \pi_{etablissementID,clientID}(ModificationAdmin)$
$b \leftarrow \pi_{etablissementID}(Commentaire)$
$c \leftarrow a * b$

$d \leftarrow \pi_{etablissementID,clientID}(c)$
$e \leftarrow a - d$
$f \leftarrow \pi_{identitifant}(e \bowtie_{clientID=ID} Utilisateur)$

6.3 Calcul relationnel tuple

6.3.1 R1

$\{u.Nom|Utilisateur(u) \wedge Commentaire(c) \wedge u.ID = c.clientID \wedge c.etablissementID >= 3 \wedge c.clientID = u.ID \wedge u.identifiant = "Brenda" \wedge c.score >= 4\}$

6.3.2 R2

$\{e.Nom|Etablissement(e) \wedge \forall u\, Utilisateur(u) \rightarrow (\exists c(Commentaire(c) \wedge e.ID = c.etablissementID \wedge c.clientID = u.ID \wedge c.score > 3 \wedge u.Nom = "Brenda"))\}$

6.3.3 R3

$\{e.Nom|Etablissement(e) \wedge \nexists c_1(Commentaire(c_1) \wedge c_1.etablissementID = e.ID) \vee \exists! c_2(Commentaire(c_2) \wedge c_2.etablissemendID = e.ID)\}$

6.3.4 R4

$\{u.Nom|Utilisateur(u) \wedge \exists e \nexists c(Etablissement(e) \wedge Commentaire(c) \wedge c.clientID = u.ID \wedge c.etablissementID = e.ID)\}$

7 Explication du fonctionnement de notre plateforme

Tout d'abord, commençons par la création de la base de donnée. Pour le script de création il n'y a rien de compliqué, ce ne sont que des requete d'insertion(on peut le voir à la section 5). Nous arrivons donc au parseur qui va commencer par le fichier "Restaurant.xml" et va le décomposer de manière à remplir la base de donne et ainsi de suite pour les fichier "Cafes.xml" et "Hotel.xml". Lors du traitement de celui-ci, les utilisateur vont être ajouté en fonction de leur activité. S' ils ont ajoute un établissement ils seront admin ou s' ils ont apposer un commentaire ou un label ils seront utilisateur. Ils

faut biensur noter que si on s'aperçoit qu'un utilisateur a créer par la suite un établissement, il va devenir un admin. Pour un établissement c'est un peu plus complexe, on va d'abord insérer l'établissement dans la table Etablissement, et puis en fonction du type de celui ci, on va faire une insertion dans la table correspondante (Restaurant,Bar ou Hotel). Il faut tout de même ne pas oublier que chaque établissement est créer par un admin, cette création sera stockée dans la table ModificationAdmin ainsi que la date de création qui sera elle aussi déterminé depuis les fichier xml. Attention, à chaque fois qu'on fais une insertion dans la table ModificationAdmin on va regarder si la date de création est bien plus récente que la date de création de l'utilisateur, sinon on fait un swap des deux dates. Passons donc aux Commentaires et aux labels. Au fur et à mesure que le parseur rencontre des commentaires ou des tags, il va ajouter les utilisateurs s'ils ne sont pas déjà existant, insérer les commentaires et leur score dans la table Commentaire et les tags dans la table Label. Idem que pour la création des établissement, on va vérifier pour chaque commentaire si la date de celui ci est plus récente que celle de la création de l'utilisateur sinon l'échange aura lieu.

Ensuite, l'interface graphique. Celle ci est très simple, elle contient une page de garde, un moteur de recherche, une page où on peut consulter notre profil si on est connecté et ensuite une page spécialement faite pour les requête demandé. Pour la page d'accueil, on peut voir les dernier établissement ajouté, pour chacun on peut le consulter, c'est à dire voir tous ce qui le défini (cfr. plus bas). La page du moteur de recherche nous permet de rechercher un établissement par le type, le nom, la commune et par un tag. Une recherche doit être au minimum composé par un de ces attribut si pas tous. Chaque combinaison est possible. Le résultat de la recherche sera affiché en dessous du moteur de recherche et pour chaque établissement nous allons pouvoir le consulter. Lors de la consultation de chaque établissement dans son contexte, on peut voir son nom ses caractéristique ainsi que sa localisation, les commentaire qui ont déjà été fait, une moyenne des score pour l'évaluer, les tags ainsi que leur poids et biensur c'est sur cette page qu'un utilisateur connecté pourra ajouter un commentaire et un tag. Nous arrivons donc à la page du profil, mais pour cela il faut bien être connecté et donc être membre (si pas membre il est possible de créer un compte). Le client peut modifier son mot de passe ainsi que son email, voir son historique d'activité qui est composé de commentaires et de tags, ainsi que pour chaque d'eux il peut les voir dans leur contexte, c'est a dire sur la page

de présentation de l'établissement expliqué plus haut. Si l'utilisateur qui est connecte est aussi admin il a la possibilité d'ajouter un établissement, le modifier ou bien en supprimer un. L'ajout est très simple, l'admin doit compléter les champs nécessaire à la description d'un établissement et choisir le type et en fonction de ce type l'utilisateur sera dirigé vers une page avec les caractéristique spécifique de ce type d'établissement où il pourra finaliser l'ajout. Pour la modification le principe est le même, l'admin peut voir tous les établissement qui sont présent et grâce à l'id d'un établissement il peut modifier seulement les champs qu'il désire pour l'établissement choisi et une fois cela fait, grâce au type de l'établissement il sera dirigé vers une autre page pour modifier les caractéristique spécifique Nous arrivons donc à la suppression d'un établissement, sur cette page l'admin peut consulter tout les établissement qui sont présent dans la base de donne et grâce à un id il peut efface l'établissement correspondant à cet id. La dernière page qui est celle des requête est une page spécialement faite pour la démonstration des requêtes SQL demandées.

Enfin, la base de notre site est la base de donné Eureka, c'est la que nous avons toute les informations. Nous ne stockons rien du cote client sauf les variable de session. Le fonctionnement de notre application est due aux variable de session et a l'url grâce au quel on peut changer de page et arriver sur celle désiré.

8 Demonstration

Nous allons supposer que la connexion avec le serveur mysql est déjà fonctionnelle. Nous allons donc creer la base de donne grace au script "CreateDB.php". Voici un output une fois que la base de donné a été créer avec succès :

```
Creating Database status: Database created successfully
Creating table Utilisateur status: Table created successfully
Creating table Etablissement status: Table created successfully
Creating table Bar status: Table created successfully
Creating table Hotel status: Table created successfully
Creating table Restaurant status: Table created successfully
Creating table ModificationAdmin status: Table created successfully
Creating table Label status: Table created successfully
Creating table Commentaire status: Table created successfully
```

Ensuite nous allons exécuter le fichier "parser.php". Voici un petit exemple :

```
CONNEXION SUCCESS
Lecture Fichier XML
*** LECTURE RESTAU XML ***
Date création : 2/10/2008
Admin : Boris
ADMIN DOESN'T EXIST
INSERT UTILISATEUR ADMIN SUCCESS
GET ID ADMIN SUCCESS 1
VERIFICATION DATECREATION UTILISATEUR ADMIN
DATECREATION UTILISATEUR ADMIN OLDEST
Name : Chez Théo Sodexho ULB
Street : Avenue Paul Héger
Num : 22
Zip : 1050
City : Ixelles
Longitude : 4.381571
Latitude : 50.813257
Tel : 02/650 49 35
Site : http://wwwdev.ulb.ac.be/restaurants/solbosch_s/r_pub.php3
ETABLISSEMENT DOESN'T EXIST
INSERT ETABLISSEMENT SUCCESS
```

Pendant que le parser s'exécute on peut très bien regarder dans la base de donne qui va progressivement se remplir avec nos établissement. Une fois celle ci fini on peut passer à l'interface graphique.

Nous arrivons à notre plateforme. Nous arrivons sur notre page de garde. Ici on va se connecter, pour cela il suffit d'aller dans la section "connexion" et une page avec un nom d'utilisateur et un mot de passe s'afficher. Si on a pas d'utilisateur on va en créer un, pour cela il y a le bouton création de compte. La création de compte se passe exactement comme pour une connexion, le client devra choisir un identifiant,un mot de passe et donner son adresse mail. Une fois connecté on va se retrouver sur la page d'accueil et on peut commencer la démo. Il faut tout de meme noter que pour voir les établissement il ne faut pas être nécessairement connecté. Voici ce que ça donne jusqu'ici :

```
Creating Database status: Database created successfully
Creating table Utilisateur status: Table created successfully
Creating table Etablissement status: Table created successfully
Creating table Bar status: Table created successfully
Creating table Hotel status: Table created successfully
Creating table Restaurant status: Table created successfully
Creating table ModificationAdmin status: Table created successfully
Creating table Label status: Table created successfully
Creating table Commentaire status: Table created successfully
```

9 Explication et justification de nos choix

Nous avons choisi mysql tout simplement parce que nous l'avons déjà tous les deux. Nous avons choisi de créer une plateforme web parce que cela convient au mieux au projet. Pour le serveur mysql nous avons utilise une "raspberry pie" sur laquelle tourne le serveur mysql. Nous pouvons l'accéder à distance grâce à un vpn. En ce qui concerne l'interface graphique de notre plateforme, nous ne nous sommes pas trop attardé la dessus puisque cela sort du cadre de ce cours, nous avons donc opter pour le choix d'un template que nous avons quelque peu modifié à notre sauce. Comme langage intermédiaire entre l'interface graphique (html et css) nous avons choisi le php puisque nous le connaission quelque peu.