

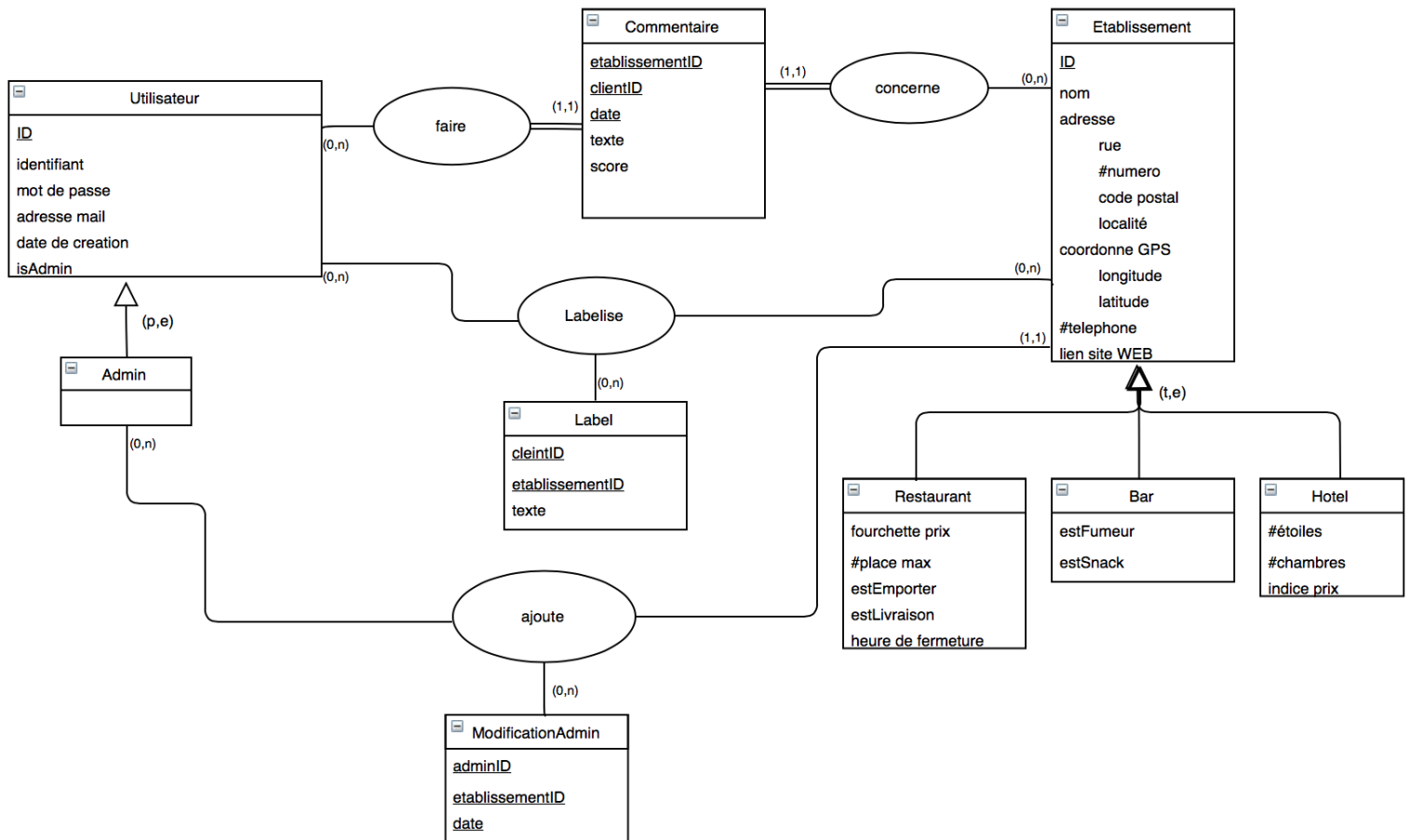
Rapport : Projet - Base de donnée

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 créer un établissement.

Les langages utilisés 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 nécessaire 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 créée 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 données en respectant les contraintes citées plus haut (date des utilisateurs). L'application est maintenant bien installée 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 voilà 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, emporteur 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 etablisementID IN (SELECT etablisementID FROM
Commentaire WHERE clientID IN (SELECT ID FROM Utilisateur WHERE
identifiant="Brenda") AND score >=4) GROUP BY clientID HAVING
count(etablisementID) >=3)
```

6.1.2 R2

```
SELECT nom FROM Etablissement WHERE ID IN (SELECT DISTINCT
etablisementID FROM Commentaire WHERE clientID IN (SELECT
DISTINCT clientID FROM Commentaire WHERE etablisementID IN
(SELECT etablisementID FROM Commentaire WHERE clientID IN
(SELECT ID FROM Utilisateur WHERE identifiant="Brenda") AND
score >=4) group by clientID having count(DISTINCT
etablisementID) >= (SELECT count(etablisementID) 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.etablisementID=e.ID) OR EXISTS
(SELECT * FROM Commentaire c WHERE c.etablisementID=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 etablisementID=m.etablisementID))
```

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

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

6.2.2 R2

$$\begin{aligned}
a &\leftarrow \pi(\sigma_{Identifiant="Brenda"}(Utilisateur)) \\
b &\leftarrow a * Commentaire \\
c &\leftarrow \sigma_{score > 3}(b) \\
d &\leftarrow \sigma_{score > 3}(Commentaire) \\
e &\leftarrow \pi_{etablissementID, clientID}(d - c) \\
f &\leftarrow e \div c \\
g &\leftarrow \pi_{Nom}(f \bowtie_{etablissementID=ID} Etablissement)
\end{aligned}$$

6.2.3 R3

$$\begin{aligned}
a &\leftarrow \pi_{ID}(Etablissement) \\
b &\leftarrow \sigma_{etablissementID \text{ NOT IN } (a) \text{ OR } \#clientID \leq 1}(Commentaire)
\end{aligned}$$

6.2.4 R4

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

$$\begin{aligned}
d &\leftarrow \pi_{\text{etablissementID}, \text{clientID}}(c) \\
e &\leftarrow a - d \\
f &\leftarrow \pi_{\text{identifiant}}(e \bowtie_{\text{clientID}=\text{ID}} \text{Utilisateur})
\end{aligned}$$

6.3 Calcul relationnel tuple

6.3.1 R1

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

6.3.2 R2

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

6.3.3 R3

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

6.3.4 R4

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

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 requête 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 donnée et ainsi de suite pour les fichiers "Cafes.xml" et "Hotel.xml". Lors du traitement de celui-ci, les utilisateurs vont être ajoutés en fonction de leur activité. S'ils ont ajouté un établissement ils seront admin ou s'ils ont posé un commentaire ou un label ils seront utilisateur. Ils

faut bien sur noter que si on s'aperçoit qu'un utilisateur a créé 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 créé par un admin, cette création sera stockée dans la table ModificationAdmin ainsi que la date de création qui sera elle aussi déterminée depuis le fichier xml. Attention, à chaque fois qu'on fait 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 existants, insérer les commentaires et leur score dans la table Commentaire et les tags dans la table Label. Idem que pour la création des établissements, 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 requêtes demandées. Pour la page d'accueil, on peut voir le dernier établissement ajouté, pour chacun on peut le consulter, c'est à dire voir tout ce qui le définit (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 composée par un de ces attributs 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 caractéristiques ainsi que sa localisation, les commentaires qui ont déjà été faits, une moyenne des scores pour l'évaluer, les tags ainsi que leur poids et bien sûr 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 à dire sur la page

de présentation de l'établissement expliqué plus haut. Si l'utilisateur qui est connecté 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 nécessaires 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 caractéristiques spécifiques 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 établissements qui sont présents 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 caractéristiques spécifiques. Nous arrivons donc à la suppression d'un établissement, sur cette page l'admin peut consulter tous les établissements qui sont présents dans la base de données et grâce à un id il peut effacer l'établissement correspondant à cet id. La dernière page qui est celle des requêtes est une page spécialement faite pour la démonstration des requêtes SQL demandées.

Enfin, la base de notre site est la base de données Eureka, c'est là que nous avons toutes les informations. Nous ne stockons rien du côté client sauf les variables de session. Le fonctionnement de notre application est due aux variables de session et à l'url grâce auquel on peut changer de page et arriver sur celle désirée.

8 Demonstration

Nous allons supposer que la connexion avec le serveur mysql est déjà fonctionnelle. Nous allons donc créer la base de données grâce au script "Create-DB.php". Voici un output une fois que la base de données a été créée avec succès :

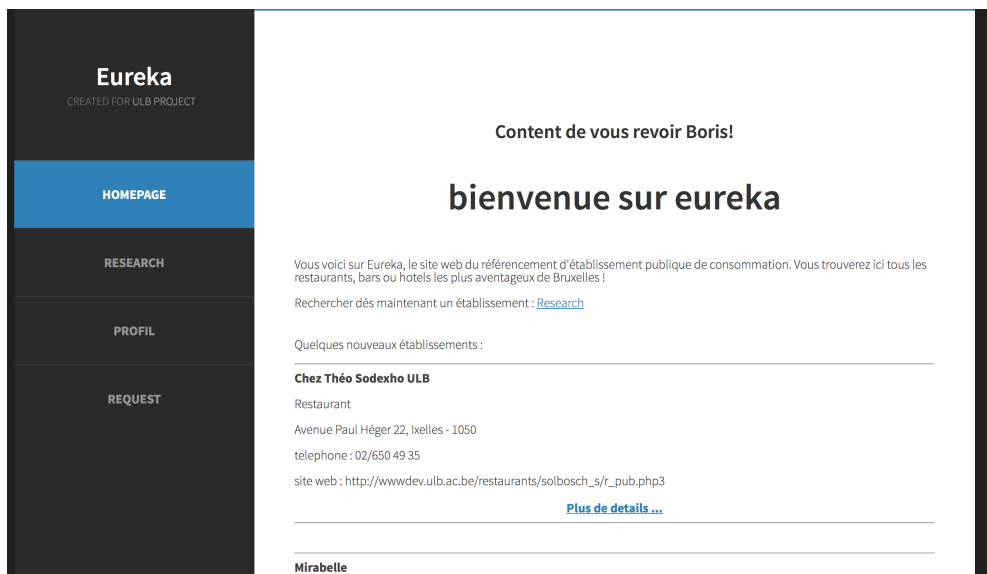
```
Creating Database status: Database created successfully
Creating table Utilisateur status: Table created successfully
Creating table Etablissement status: Table created successfully
Creating table Bar status: Table created successfully
Creating table Hotel status: Table created successfully
Creating 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 :

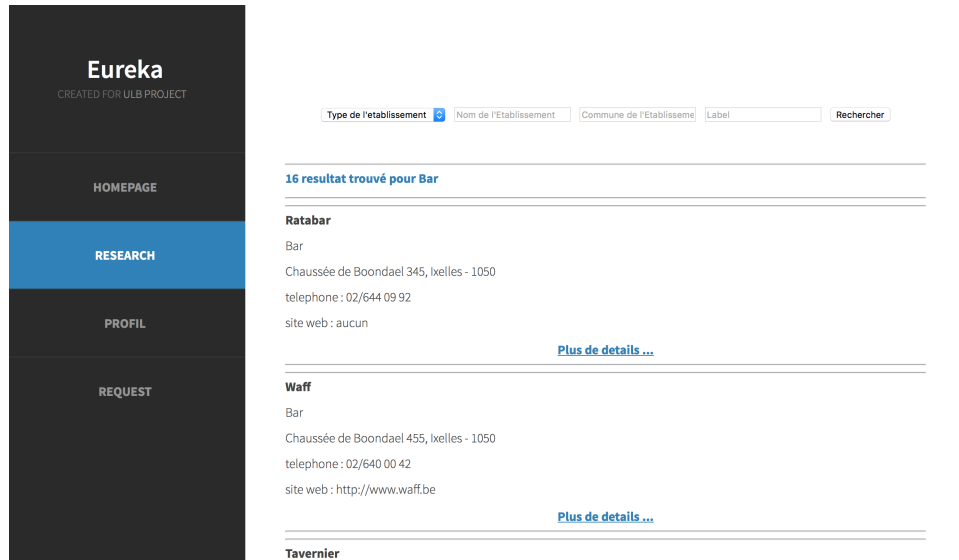
```
CONNEXION SUCCESS
Lecture Fichier XML
*** LECTURE RESTAU XML ***
Date création : 2/10/2008
Admin : Boris
ADMIN DOESN'T EXIST
INSERT UTILISATEUR ADMIN SUCCESS
GET ID ADMIN SUCCESS 1
VERIFICATION DATECREATION UTILISATEUR ADMIN
DATECREATION UTILISATEUR ADMIN OLDEST
Name : Chez Théo Sodexho ULB
Street : 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.

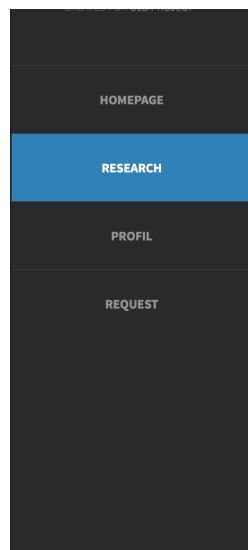
Nous arrivons à notre plateforme. Nous arrivons sur notre page de garde. Ici on va se connecter, pour cela il suffit d'aller dans la section "connexion" et une page avec un nom d'utilisateur et un mot de passe s'afficher. Si on a pas d'utilisateur on va en créer un, pour cela il y a le bouton création 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 :



Nous allons maintenant essayer le moteur de recherche, nous allons rechercher les établissements qui sont des bars. Voyons ce que ça donne :



Pour chaque résultat si on clique sur le lien "Plus de détails" on va arriver sur la page de présentation d'un établissement. Regardons :



ratabar

Informations sur leurs services:

Smoking : Oui

Snack : Non

Localisation:

Chaussée de Boondael 345, Ixelles - 1050

Coordonnées : 50.8186 - 4.38407



Contact:

Téléphone : 02/644 09 92

Site web : Aucun

C'est aussi sur cette page que le client va pouvoir commenter et apposer un tag tout en respectant les contraintes vu plus haut. Voici un exemple de commentaire :

Commentaires:

Brenda
2008-08-07
Score : ★★☆☆☆
Spécialité : sous marin. Petit défaut, il n'y a rien à manger mais on peut aller chercher un durum en face.

Boris
2008-05-02
Score : ★★★★★
Bonne ambiance à partir de 23h. Déco apache. Spécialité : sous marin.

Moyenne des scores :
★★★★☆

Ajouter un commentaire :
Score : 4

J'aime l'ambiance ici

Commenter

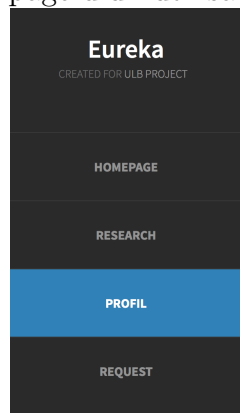
Voici un exemple de tag :

Tags de l'établissement:

Label	Apposé
Bon marché	3
Fumeur	3
Babyfoot	1
Service au bar	1

Ajouter un tag :

On va avancer et on va arriver à la page du profil de l'identifiant, voici une page d'un utilisateur et d'un admin :



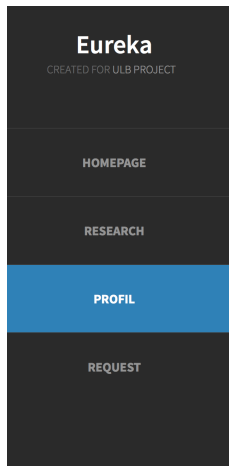
Profile de george

Mes information :

Identifiant	Mot de passe	Email	Date de Creation du compte
george	*****	George@email.com	1996-11-15

Action :

[Se deconnecter](#)
[Modifier mot de passe](#)
[Modifier email](#)
[Voir historique de mon activité](#)



Profile de boris

Mes information :

Identifiant	Mot de passe	Email	Date de Creation du compte
boris	*****	Boris@email.com	2003-04-04

Action :

[Se deconnecter](#)
[Modifier mot de passe](#)
[Modifier email](#)
[Voir historique de mon activité](#)
[Ajouter un etablissement](#)
[Modifier un etablissement](#)
[Supprimer un etablissement](#)

Comme vous pouvez le voir l'admin qui est dans notre exemple Boris peut ajouter, modifier et supprimer des établissements tandis qu'un utilisateur ne

détient pas ce pouvoir.

Voyons comment se passe l'ajout d'un établissement :

Eureka

CREATED FOR ULB PROJECT

HOME PAGE

RESEARCH

PROFIL

REQUEST

Ajouter un etablissement

Nom de l'etablissement :

Rue :

Numero :

Code Postal :

Localite :

Longitude :

Latitude :

Telephone :

Site :

Type :

Une fois cette première étape effectuée, nous arrivons sur cette page :

Eureka

CREATED FOR ULB PROJECT

HOME PAGE

RESEARCH

PROFIL

REQUEST

Ajouter un etablissement

Prix :

Nombre de place pour un Banquet :

Emporter : ☒ Oui ☐ Non

Livraison : ☐ Oui ☒ Non

Ouverture :

Lundi :	<input type="radio"/> Toute la journée	<input type="radio"/> Fermé toute la journée	<input checked="" type="radio"/> Matin	<input type="radio"/> Après-midi
Mardi :	<input checked="" type="radio"/> Toute la journée	<input type="radio"/> Fermé toute la journée	<input type="radio"/> Matin	<input type="radio"/> Après-midi
Mercredi :	<input checked="" type="radio"/> Toute la journée	<input type="radio"/> Fermé toute la journée	<input type="radio"/> Matin	<input type="radio"/> Après-midi
Jedi :	<input type="radio"/> Toute la journée	<input type="radio"/> Fermé toute la journée	<input type="radio"/> Matin	<input checked="" type="radio"/> Après-midi
Vendredi :	<input type="radio"/> Toute la journée	<input checked="" type="radio"/> Fermé toute la journée	<input type="radio"/> Matin	<input type="radio"/> Après-midi
Samedi :	<input checked="" type="radio"/> Toute la journée	<input type="radio"/> Fermé toute la journée	<input type="radio"/> Matin	<input type="radio"/> Après-midi
Dimanche :	<input type="radio"/> Toute la journée	<input checked="" type="radio"/> Fermé toute la journée	<input type="radio"/> Matin	<input type="radio"/> Après-midi

Passons maintenant à la modification d'un établissement :

25	Suprabailly	Rue du bailli	77	1050	Ixelles	4.3601	50.8258	02/534 09 01	
26	SuperMercado	Rue de Florence	100	1050	Ixelles	4.36018	50.8296	02/222 22 22	
27	Brasserie L'union	Parvis de Saint-Gilles	55	1050	Ixelles	4.34608	50.8308	02/538 15 79	
28	Greenwich	Rue des Chartreux	5	1050	Ixelles	4.34748	50.8488	02/511 41 67	
29	Michael Collins	Rue du Bailli	1	1050	Ixelles	4.36349	50.8272	02/644 61 21	
30	de Valera's	Place Flagey	17	1050	Ixelles	4.37169	50.8275	02/649 80 54	
31	Sofitel	Avenue Louise	42	1180	Bruxelles	4.35693	50.8363	02/356 23 78	http://www.so
32	Hilton	Gare central	1	1180	Bruxelles	4.35603	50.846	02/111 22 33	http://www.hil

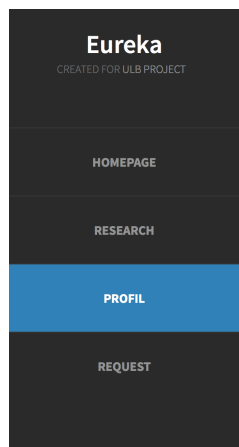
Choisir un etablissement

Veillez choisir l'etablissement avec l'id et completer uniquement les colonnes que vous voulez modifier

ID	Nom	Rue	Numero	CodePostal
32	hotel de Luxe			

Modifier Annuler

Il faut préciser que l'admin peut voir tout les établissements existant dans la base de données et en choisissant un id, il peut modifier les colonnes qu'il désire de l'établissement correspondant. Même principe que pour l'ajout, nous arrivons sur une autre page :



Modifier un etablissement

Prix	Nombre de chambres	Nombre d'etolles'
1000	100	4

Veillez remplir les colonnes que vous voulez modifier

Prix	Nombre de chambres	Nombre d'etolles'

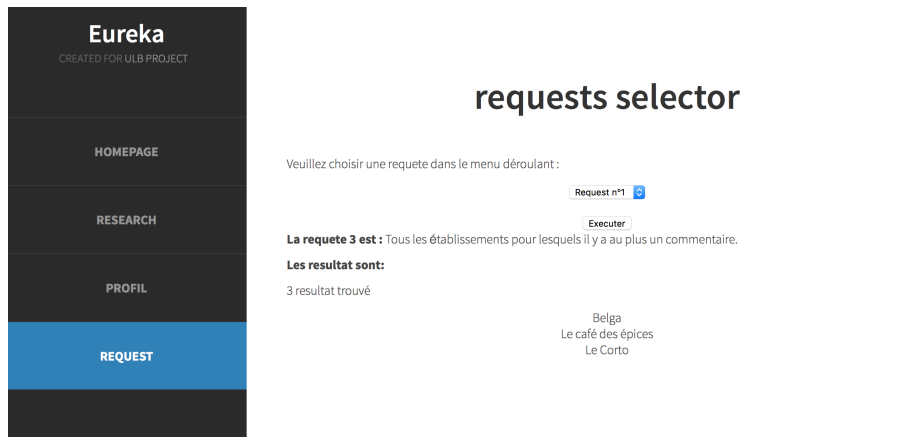
Update Annuler

© EUREKA. ALL RIGHTS RESERVED.
DESIGN BY TEMPLATED.

Encore une fois l'admin peut voir les caractéristiques de l'établissement et peut ou non modifier certaines informations.

La suppression se fait avec le même principe, la seule différence c'est que l'admin n'a qu'une seule étape, le choix de l'établissement à supprimer qui se fait comme pour la modification grâce à un id.

Pour terminer notre bref démonstration, la page des requêtes :



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 connaissons quelque peu.