

INFO-F-303 Base de données
Projet : Annuaire d'établissements horeca
Rapport de deuxième partie

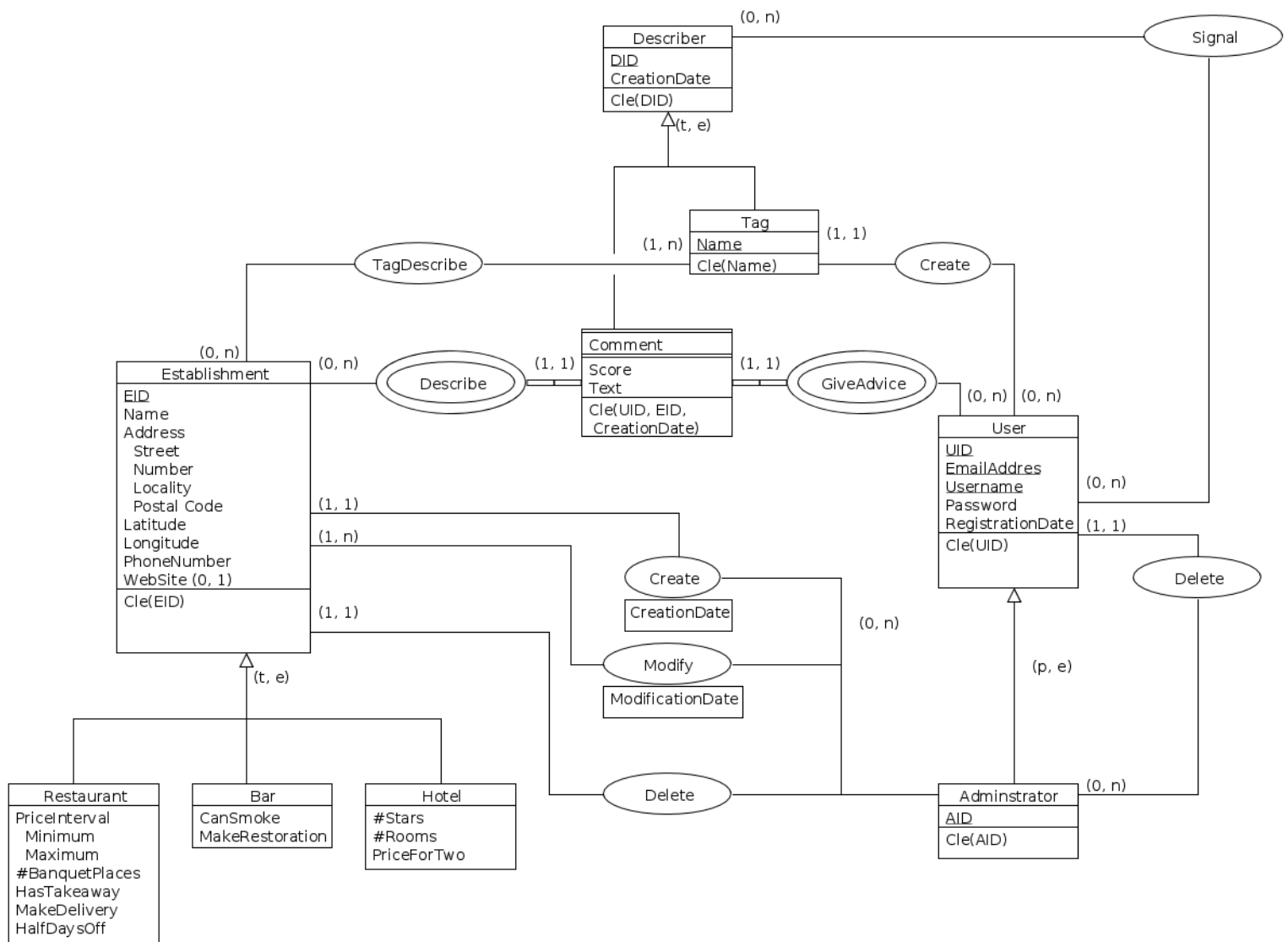
Hakim Boulahya & Youcef Bouharaoua

Mai 2016

Contents

1	Modèle entité-association	2
2	Contraintes d'intégrité	2
3	Modèle relationnel	3
4	Script SQL DDL	4
5	Requête 1	7
5.1	SQL	7
5.2	Algèbre relationnelle	7
5.3	Calcul relationnel	7
6	Requête 3	8
6.1	SQL	8
6.2	Algèbre relationnelle	8
6.3	Calcul relationnel	8
7	Requête 4	8
7.1	SQL	8
7.2	Algèbre relationnelle	8
7.3	Calcul relationnel	8
8	Requête 5	8
8.1	SQL	8
9	Requête 6	9
9.1	SQL	9
10	Application	9
10.1	Installation	9
10.2	Démonstration	10

1 Modèle entité-association



2 Contraintes d'intégrité

- L'*UID* Et L'*EID* sont respectivement unique à chaque *User* et *Establishement*.
- *User.EmailAdress* est unique.
- Le *User* doit impérativement être connecté pour pouvoir effectuer un *Comment*.
- *Comment.Score* varie entre 0 et 5.
- *Comment.Text* est fixé à maximum 200 caractères .
- La taille de *User.Password* varie entre 4 et 16 caractères.
- Le *User* a le droit d'écrire un *Comment* par jour par *Establishement*.

- Le *User* ne peut apposer qu'une seule fois le même *Tag* sur un même *Establishement*.
- Un *User* a le droit de signaler un *Describer* d'un autre *User*, si celui-ci utilise un langage inadéquat ou si le commentaire porte des propos racistes ou injurieux.
- L' *Administrator* a la possibilité de supprimer un *User* si nécessaire.
- Tous les *Administrator* peuvent modifier ou supprimer n'importe quel *Establishment*.

3 Modèle relationnel

- Establishment(EID, Name, Latitude, Longitude, PhoneNumber, Modified, *WebSite*)
 - Restaurant.EID reference Etablissment.EID.
- Restaurant(EID, PriceMinimum, PriceMaximum, BanquetPlaces, HasTakeaway, MakeDelivery, HalfDaysOff)
 - Restaurant.EID reference Etablissment.EID.
- Bar(EID, CanSmoke, MakeRestoration)
 - Bar.EID reference Etablissment.EID.
- Hotel(EID, NoStars, NoRooms, PriceForTwo)
 - Hotel.EID reference Etablissment.EID.
- Address(EID, Street, No, Locality, PostalCode)
 - Address.EID reference Etablissment.EID.
- User(UID, EmailAddress, Username, Password, RegistrationDate)
- Administrator(UID, AID)
 - Administrator.UID fait référence a User.UID.
- EstablishmentCreation(EID, AID, CreationDate)
 - EstablishmentCreation.EID référence Establishment.EID.
 - EstablishmentCreation.AID référence Administrator.AID.
- EstablishmentModification(OldEID, NewEID, AID, ModificationDate)
 - EstablishmentModification.OldEID référence Establishment.EID.
 - EstablishmentModification.NewEID référence Establishment.EID.
- EstablishmentDeletion(EID, AID, DeletionDate)
 - EstablishmentDeletion.EID référence Establishment.EID.
 - EstablishmentDeletion.AID référence Administrator.AID.
- UserDeletion(UID, AID, DeletionDate)
 - UserDeletion.UID référence User.UID.
 - UserDeletion.AID référence Administrator.AID.

- Describer(DID)
- Comment(DID, UID, EID, CreationDate, Score, Text)
 - Comment.DID référence Describer.DID.
- Tag(DID, Name, UID, CreationDate)
 - Tag.DID référence Describer.DID.
 - Tag.UID référence User.UID.
- Signal(DID, SignalerUID)
- TagDescribe(Name, EID, UID)
 - TagDescribe.Name référence Tag.Name.
 - TagDescribe.EID référence Establishment.EID.
 - TagDescribe.UID référence User.UID.

Remarques

- Un *Establishment* ne peut pas exister sans qu'il soit associé à un *Restaurant*, *Bar* ou *Hotel*.
- un *User.UID* n'existant pas dans une *Administrator.UID* est un simple utilisateur n'ayant pas de droit de création/modification/suppression d'un établissement ou d'un *User*.
- La modification d'un *Establishment* étant possible, et que nous voulons garder les traces de modifications, il a été décidé que pour chaque modification un nouvelle *Establishment* sera créé. Pour savoir si un *Establishment* a été modifié il suffit de voir son champ *Establishment.Modified*. Si cette valeur est à *true*, c'est qu'il a été modifié et qu'il suffit donc de récupérer le nouvelle *Establishment.EID* via la relation *EstablishmentModification*. *Establishment.OldEID* référence l'*Establishment* ayant été modifié, et *Establishment.NewEID* référence l'*Establishment* où les modifications ont été apporté.
- *EstablishmentModification.OldEID* et *EstablishmentModification.NewEID* ne peuvent pas être identique.
- *UserDeletion.UID* ne peut pas référencer le *Administrator.UID* associé à *UserDeletion.AID*
- DID doit être unique entre les entités *Comment* et *Tag*.

4 Script SQL DDL

```

1 # Create a user too access to brusselsbook database
2 CREATE USER 'bbadmin'@'localhost' IDENTIFIED BY 'common';
3
4 # Create the database
5 CREATE DATABASE brusselsbook CHARACTER SET 'utf8';
6
7 # Give access to the user
8 GRANT ALL PRIVILEGES ON 'brusselsbook'.* TO 'bbadmin'@'localhost';
9
10 # Use the new created database
11 USE brusselsbook;
12
```

```

13 CREATE TABLE Establishment(
14     EID INT UNSIGNED NOT NULL AUTO.INCREMENT,
15     EName VARCHAR(50) NOT NULL,
16     PhoneNumber VARCHAR(20) NOT NULL,
17     Modified TINYINT(1) DEFAULT 0,
18     Website VARCHAR(200),
19     Type INT UNSIGNED NOT NULL,
20     PRIMARY KEY(EID)
21 );
22
23 CREATE TABLE Restaurant(
24     EID INT UNSIGNED NOT NULL,
25     PriceRange INT UNSIGNED NOT NULL,
26     BanquetPlaces INT UNSIGNED NOT NULL,
27     HasTakeaway TINYINT(1) NOT NULL,
28     MakeDelivery TINYINT(1) NOT NULL,
29     # String format : XXXXXXXXXXXXXXX where X can be F(false) or T(true)
30     # To know if the restau. is open in the half day
31     HalfDaysOff VARCHAR(14) NOT NULL,
32     FOREIGN KEY (EID) REFERENCES Establishment(EID)
33     ON DELETE CASCADE
34 );
35
36 CREATE TABLE Bar(
37     EID INT UNSIGNED NOT NULL,
38     CanSmoke TINYINT(1) NOT NULL,
39     MakeRestoration TINYINT(1) NOT NULL,
40     FOREIGN KEY (EID) REFERENCES Establishment(EID)
41     ON DELETE CASCADE
42 );
43
44 CREATE TABLE Hotel(
45     EID INT UNSIGNED NOT NULL,
46     NoStars SMALLINT UNSIGNED NOT NULL,
47     NoRooms SMALLINT UNSIGNED NOT NULL,
48     PriceForTwo FLOAT(10, 2) NOT NULL,
49     FOREIGN KEY (EID) REFERENCES Establishment(EID)
50     ON DELETE CASCADE
51 );
52
53 CREATE TABLE Address(
54     EID INT UNSIGNED NOT NULL,
55     Street VARCHAR(50) NOT NULL,
56     StreetNumber VARCHAR(10) NOT NULL,
57     Locality VARCHAR(50) NOT NULL,
58     PostalCode VARCHAR(10) NOT NULL,
59     Latitude FLOAT(10, 4) NOT NULL,
60     Longitude FLOAT(10, 4) NOT NULL,
61     FOREIGN KEY (EID) REFERENCES Establishment(EID)
62     ON DELETE CASCADE
63 );
64
65 CREATE TABLE BookUser(
66     UID INT UNSIGNED NOT NULL AUTO.INCREMENT,
67     EmailAddress VARCHAR(50) NOT NULL UNIQUE KEY,
68     Username VARCHAR(15) NOT NULL UNIQUE KEY,
69     Pwd VARCHAR(16) NOT NULL,
70     RegistrationDate DATETIME DEFAULT CURRENT.TIMESTAMP,
71     PRIMARY KEY(UID)
72 );
73
74 CREATE TABLE Administrator(
75     AID INT UNSIGNED NOT NULL AUTO.INCREMENT,
76     UID INT UNSIGNED NOT NULL,
77     PRIMARY KEY(AID),

```

```

78 FOREIGN KEY (UID) REFERENCES BookUser(UID)
79 ON DELETE CASCADE
80 );
81
82 # Even if Describer is not in the relationnal model,
83 # we choose to create this table (less difficult to manage the modifications)
84 CREATE TABLE Describer(
85     DID INT UNSIGNED NOT NULL AUTOINCREMENT,
86     PRIMARY KEY(DID)
87 );
88
89 CREATE TABLE BookComment(
90     DID INT UNSIGNED NOT NULL,
91     UID INT UNSIGNED NOT NULL,
92     EID INT UNSIGNED NOT NULL,
93     CreationDate DATETIME NOT NULL DEFAULT NOW() ,
94     Score TINYINT UNSIGNED NOT NULL,
95     BookText TEXT NOT NULL,
96     UNIQUE KEY(DID, UID, CreationDate),
97     FOREIGN KEY (DID) REFERENCES Describer(DID)
98     ON DELETE CASCADE,
99     FOREIGN KEY (UID) REFERENCES BookUser(UID)
100     ON DELETE CASCADE,
101     FOREIGN KEY (EID) REFERENCES Establishment(EID)
102     ON DELETE CASCADE
103 );
104
105 CREATE TABLE Tag(
106     DID INT UNSIGNED NOT NULL,
107     TagName VARCHAR(50) NOT NULL UNIQUE KEY,
108     UID INT UNSIGNED NOT NULL,
109     CreationDate DATETIME DEFAULT CURRENT_TIMESTAMP,
110     FOREIGN KEY (DID) REFERENCES Describer(DID)
111     ON DELETE CASCADE,
112     FOREIGN KEY (UID) REFERENCES BookUser(UID)
113     ON DELETE CASCADE
114 );
115
116 CREATE TABLE TagDescribe(
117     TagName VARCHAR(50) ,
118     EID INT UNSIGNED NOT NULL,
119     UID INT UNSIGNED NOT NULL,
120     FOREIGN KEY (TagName) REFERENCES Tag(TagName)
121     ON DELETE CASCADE,
122     FOREIGN KEY (EID) REFERENCES Establishment(EID)
123     ON DELETE CASCADE,
124     FOREIGN KEY (UID) REFERENCES BookUser(UID)
125     ON DELETE CASCADE
126 );
127
128 CREATE TABLE UserSignal(
129     DID INT UNSIGNED NOT NULL,
130     SignalerUID INT UNSIGNED NOT NULL,
131     FOREIGN KEY (DID) REFERENCES Describer(DID)
132     ON DELETE CASCADE,
133     FOREIGN KEY (SignalerUID) REFERENCES BookUser(UID)
134     ON DELETE CASCADE
135 );
136
137 CREATE TABLE EstablishmentCreation(
138     EID INT UNSIGNED NOT NULL UNIQUE KEY,
139     AID INT UNSIGNED NOT NULL,
140     CreationDate DATETIME DEFAULT CURRENT_TIMESTAMP,
141     FOREIGN KEY (EID) REFERENCES Establishment(EID)
142     ON DELETE CASCADE,

```

```

143 FOREIGN KEY (AID) REFERENCES Administrator(AID)
144 ON DELETE CASCADE
145 );
146
147 CREATE TABLE EstablishmentModification(
148     OldEID INT UNSIGNED NOT NULL,
149     NewEID INT UNSIGNED NOT NULL,
150     AID INT UNSIGNED NOT NULL,
151     ModificationDate DATETIME DEFAULT CURRENT_TIMESTAMP,
152     FOREIGN KEY (OldEID) REFERENCES Establishment(EID)
153     ON DELETE CASCADE,
154     FOREIGN KEY (NewEID) REFERENCES Establishment(EID)
155     ON DELETE CASCADE,
156     FOREIGN KEY (AID) REFERENCES Administrator(AID)
157     ON DELETE CASCADE
158 );
159
160 CREATE TABLE EstablishmentDeletion(
161     EID INT UNSIGNED NOT NULL,
162     AID INT UNSIGNED NOT NULL,
163     DeletionDate DATETIME DEFAULT CURRENT_TIMESTAMP,
164     FOREIGN KEY (EID) REFERENCES Establishment(EID)
165     ON DELETE CASCADE,
166     FOREIGN KEY (AID) REFERENCES Administrator(AID)
167     ON DELETE CASCADE
168 );
169
170 CREATE TABLE UserDeletion(
171     UID INT UNSIGNED NOT NULL,
172     AID INT UNSIGNED NOT NULL,
173     DeletionDate DATETIME DEFAULT CURRENT_TIMESTAMP,
174     FOREIGN KEY (UID) REFERENCES BookUser(UID)
175     ON DELETE CASCADE,
176     FOREIGN KEY (AID) REFERENCES Administrator(AID)
177     ON DELETE CASCADE
178 );

```

5 Requête 1

5.1 SQL

```

1 SELECT u.* FROM BookComment c, BookUser u, BookUser ub
2 WHERE (c.UID = u.UID) AND (c.Score > 3)
3 AND (u.Username <> 'Brenda') AND (ub.Username = 'Brenda')
4 AND EXISTS(SELECT c2.* FROM BookComment c2 WHERE (c2.UID = ub.UID)
5 AND (c.EID = c2.EID) AND (c2.Score > 3)) GROUP BY c.UID
6 HAVING COUNT(DISTINCT c.EID) > 2

```

5.2 Algèbre relationnelle

$$\Pi_{(U.UID)}(\sigma_{(count(cnt)>3)}(\sigma_{(Score \geq 3 \wedge UID \neq 7)}(\Pi_{(EID)}(\sigma_{(UID=7 \wedge Score \geq 3)}(Describer * BookComment) * BookComment)(name \ g \ count \ distinct(BC.EID) \ as \ cnt)))$$

5.3 Calcul relationnel

$$u.UID | BookUser(u) \wedge BookComment(b) \wedge \exists b1(b1.UID = 7 \wedge b1.score \geq 3 \wedge Count(E.EID) \geq 3) \\ \wedge \exists U(B.UID = U.UID \wedge COUNT(B.EID) \geq 3)$$

6 Requête 3

6.1 SQL

```
1 SELECT * FROM Establishment e WHERE
2 (SELECT COUNT(*) FROM BookComment c WHERE c.EID = e.EID) <= 1
```

6.2 Algèbre relationnelle

$$e.EID, b.EID | Establishment(e) \wedge BookComment(b) \wedge \forall e \rightarrow \\ \exists! B2 (BookComment(B2) \wedge B2.EID = e.EID) \vee B2 (BookComment(B2) \wedge B2.EID = e.EID)$$

6.3 Calcul relationnel

$$\Pi_{(E.EID)}(\sigma_{count(cnt)=1}(name\ g\ count\ (*)\ (BC)\ as\ cnt))$$

7 Requête 4

7.1 SQL

```
1
2 SELECT u.* FROM Administrator a, EstablishmentCreation ec, BookUser u
3 WHERE ec.AID = a.AID AND a.UID = u.UID
4 AND NOT EXISTS(SELECT * FROM BookComment c
5 WHERE c.UID = a.UID AND c.EID = ec.EID)
6 GROUP BY(ec.AID)
```

7.2 Algèbre relationnelle

Pour que l'algèbre de la requête soit claire les tables utilisées pour celle-ci seront identifiées ainsi:
BookComment B ,EstablishmentCreation EC ,Administrator A

$$\Pi_{(A.UID)}(\sigma_{(A.UID=B.UID\ AND\ EC.EID=B.EID)}(\Pi_{(UID,EID,AID)}(EC * A) * B))$$

7.3 Calcul relationnel

$$A.AID | Administrator(A) \wedge \forall A \rightarrow \\ \nexists EC \nexists B (EstablishmentCreation(EC) \wedge BookComment(B) \wedge B.UID = A.UID \wedge B.EID = EC.EID) \\ \wedge EC.AID = A.AID)$$

8 Requête 5

8.1 SQL

```
1
2 SELECT e.*, AVG(c.Score)
3 AS AvgScore FROM BookComment c, Establishment e WHERE
4 e.EID = c.EID AND c.EID IN
5 (SELECT e.EID FROM Establishment e WHERE
6 (SELECT COUNT(*) FROM BookComment c WHERE e.EID = c.EID ) >= 3 )
7 GROUP BY (c.EID) ORDER BY AvgScore
```


9 Requête 6

9.1 SQL

```
1
2 SELECT td.TagName, (SELECT AVG(c.Score)
3 FROM BookComment c, TagDescribe td2
4 WHERE (td.TagName = td2.TagName) AND (c.EID = td2.EID)) as AvgScore
5 FROM BookComment c, TagDescribe td
6 GROUP BY td.TagName HAVING COUNT(DISTINCT td.EID) > 4
7 ORDER BY AvgScore
```

10 Application

10.1 Installation

Prérequis

Pour pouvoir démarrer le projet il est nécessaire d'installer les programmes suivant:

- Java 8
- Gradle
- Apache Tomcat v8.0.33
- mysql-5.6

Nous considérons que vous utilisez un OS basé sur Unix.

Pour installer gradle sous Ubuntu:

```
1 sudo apt-get install gradle
```

Pour Apache Tomcat télécharger <http://apache.belnet.be/tomcat/tomcat-8/v8.0.33/bin/apache-tomcat-8.0.33.zip> et dézippez-le.

Démarrer le programme

Pour les commandes suivantes placez-vous à la racine du projet *i.e.* /BrusselsBook.

Pour exécuter le script SQL DDL, connectez-vous à mysql via un terminal:

```
1 mysql -u root -p
```

Après avoir entré votre mot de passe:

```
1 SOURCE script/brusselsBook.sql
```

Veillez à exécuter le script SQL DDL avant la génération.

Pour générer les données des fichiers xml:

```
1 bash runner generate
```

Pour démarrer le serveur web:

```
1 bash runner server <chemin/vers/apachetomcat>
```

Une erreur peut survenir parfois par manque de permissions, changer ces permissions:

```
1 chmod +x <chemin/vers/apachetomcat>/bin/catalina.sh
```

Pour arrêter le serveur web:

```
1 bash runner shutserver <chemin/vers/apachetomcat>
```

Après avoir démarré le serveur connecté via votre navigateur web, de préférence Firefox, à l'adresse <http://localhost:8080/BrusselsBook/home>.

Pour vous connecter, utilisez l'utilisateur **boris** et le mot de passe **common** (cette utilisateur est ajouter via le script de génération des fichiers xml).

10.2 Démonstration

En vous connectant à l'url <http://localhost:8080/BrusselsBook/home>, vous êtes sur la page d'accueil de l'application [Figure 1].

Depuis cette page il vous est soit possible de vous connecter à un compte [Figure 2] ou d'effectuer une recherche [Figure 4].

Lors de la connexion à un compte si vous n'avez pas d'identifiant, il vous est possible de cliquer sur le lien de création du compte qui vous redirigera vers la page de création d'un utilisateur [Figure 3].

Si vous effectuez une recherche, les résultats proposés [Figure 4] sont tous les établissements et les utilisateurs correspondant à votre recherche. Il vous possible d'entrée l'adresse, le code postal, la localité, le nom d'un établissement, le nom d'un utilisateur ou l'email d'un utilisateur.

Lors du clique sur un établissement vous arrivez sur la fiche de cette établissement [Figure 5]. Sur cette fiche vous accédez à toutes les informations de cette établissement. Si vous êtes connecté en tant qu'un utilisateur il vous est possible: d'ajouter un tag déjà existant, de créer un tag, de signaler un commentaire ou encore de rédiger un commentaire [Figure 6]. Si vous n'êtes pas connecté, il ne vous sera pas possible d'effectuer toutes ces actions.

Lorsque vous cliquez sur un utilisateur, vous accédez à sa fiche [Figure 7], ou vous pouvez visualisez tous les commentaires écrit par cette utilisateur.

Si vous êtes connecté en tant qu'administrateur, un lien vers la page d'administration [Figure 8] est accessible dans la barre principale. Sur cette page d'administration il vous est possible de créer des établissements (hotel, restaurant ou café) [Figure 9], de les gérer, de gérer les utilisateurs, de gérer les commentaires signalés et également de visualiser le résultat des requêtes SQL demandées.

Sur la page de gestion des établissements [Figure 10] il vous est possible de supprimer des établissements du site. Ils seront toujours disponible dans la base de données mais non visible sur le site. Il vous est possible de les replacer sur le site. Il est également possible de supprimer de la base de donnée un établissement. Il ne vous sera plus possible de récupérer cette établissement. Il est également possible de modifier un établissement. Après la modification, il est possible d'accéder à l'ancienne version de l'établissement via un lien accessible sur la fiche de l'établissement. Toutes les modifications sont enregistrées dans la base de données, mais il est possible de les supprimer définitivement via la page de gestion.

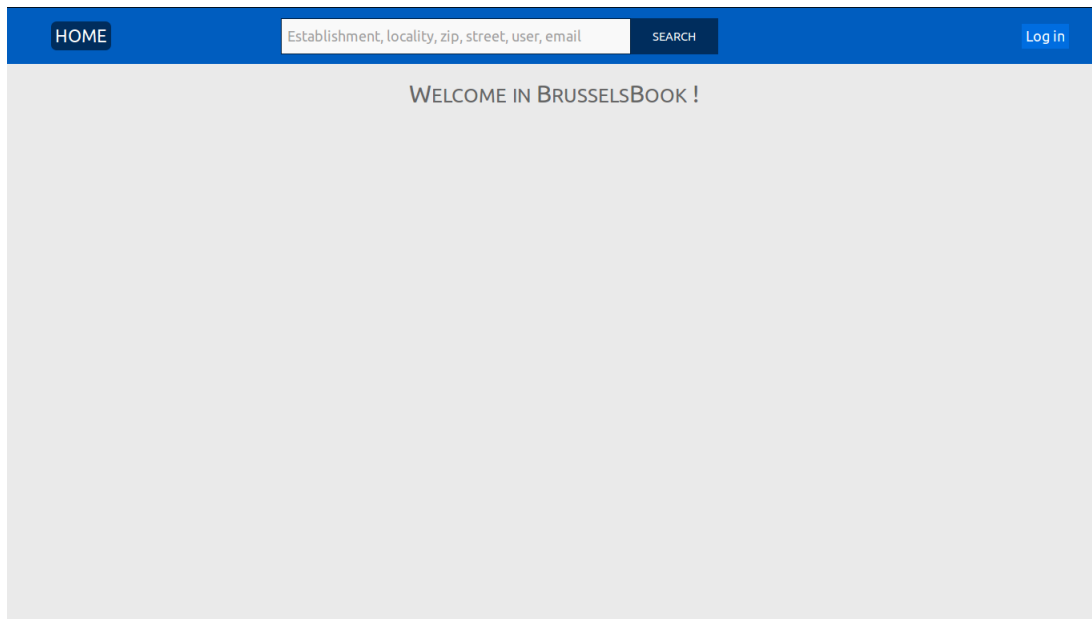


Figure 1: Page d'accueil

Sur la page de gestion des utilisateurs [Figure 11], il vous est possible soit de : donner les droits d'administration à un simple utilisateur, de supprimer un utilisateur de la base de donnée ou de désactiver un compte. Pour cette dernière action, l'utilisateur concerné ne pourra plus se connecté au site web. Il vous est possible d'annuler cette désactivation.

Sur la page de gestion des commentaires signalés, vous visualisez tous les commentaires signalés par les utilisateurs. Il vous est soit possible de supprimer de la base de données le commentaire, soit d'annuler le signalement.

HOME

Establishment, locality, zip, street, user, email

SEARCH

Log in

ENTER YOUR CREDENTIALS

Email or username

LOG IN

Don't have an account ? Create one !

localhost:8080/BrusselsBook/login

Figure 2: Page de connexion

HOME

Establishment, locality, zip, street, user, email

SEARCH

Log in

CREATE YOUR ACCOUNT

Your username

Your email

CREATE

Figure 3: Page de création d'un compte

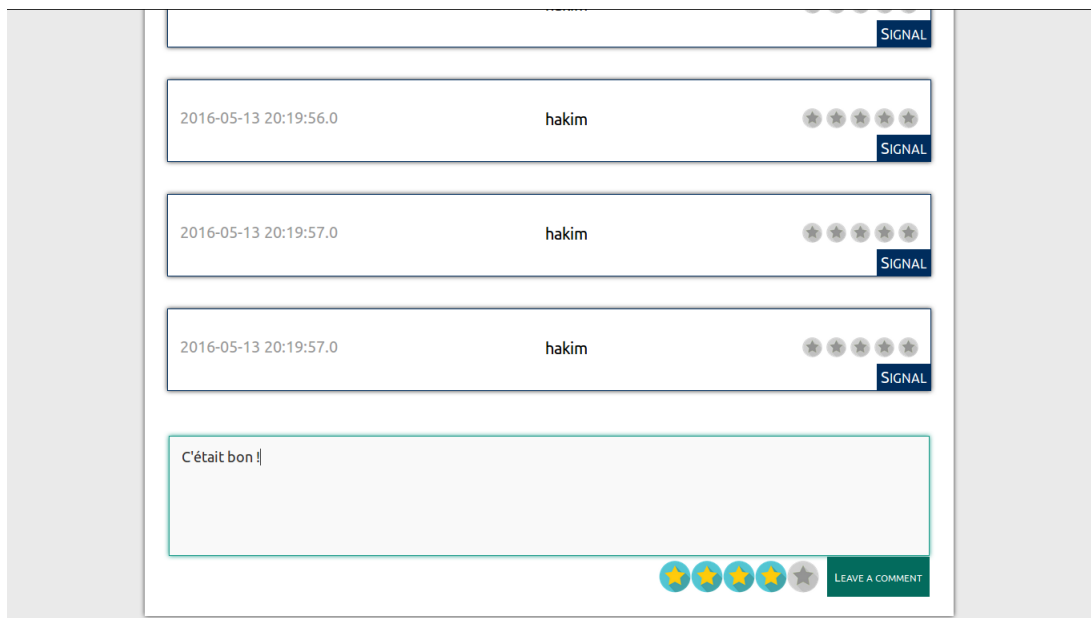


Figure 6: Commentaire sur la fiche d'un établissement

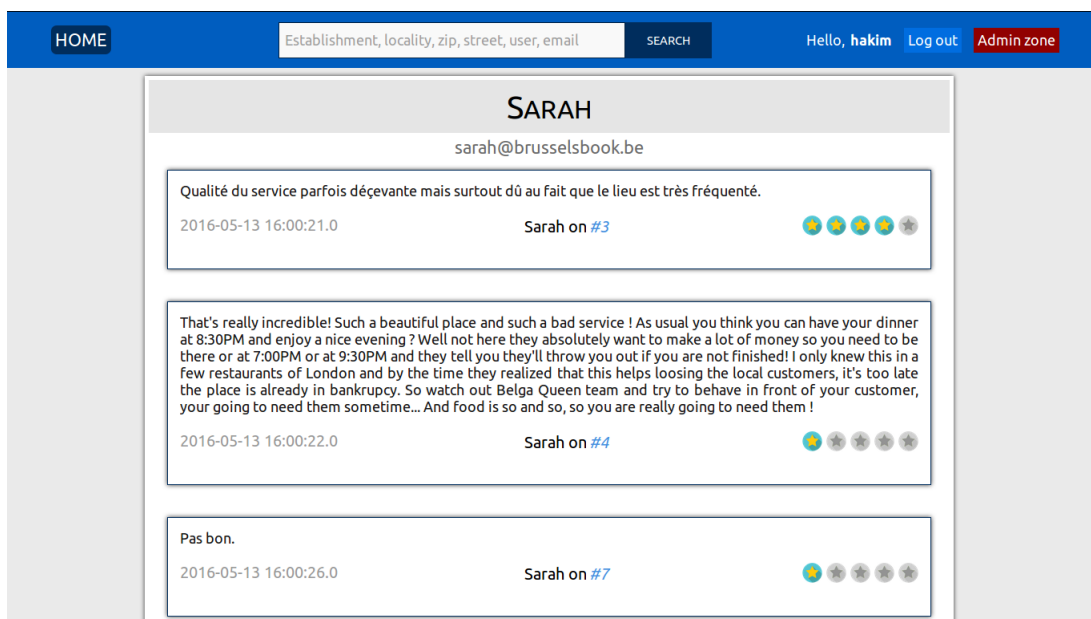


Figure 7: Fiche d'un utilisateur

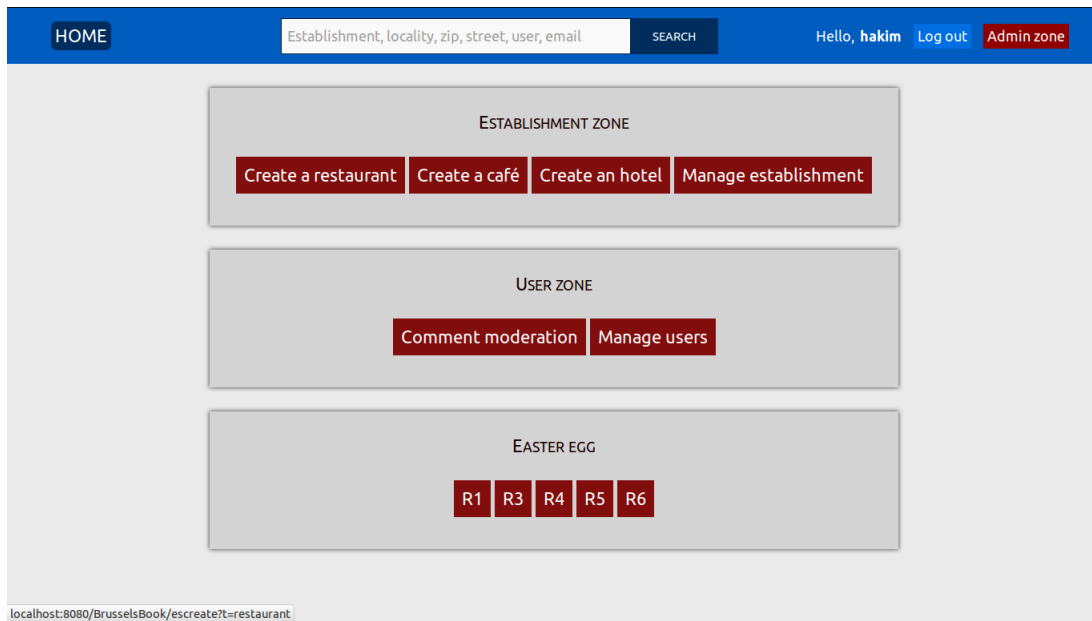


Figure 8: Page d'administration

The screenshot shows the 'CREATE RESTAURANT' form, which is a vertical stack of input fields and controls. At the top, the title 'CREATE RESTAURANT' is centered. Below it are several text input fields: 'Establishment name', 'Phone number', 'Website url', 'Street', 'Street number', 'Locality', and 'Zip code'. Following these are two numeric input fields with spinners: 'AVERAGE PRICE' and 'NUMBER OF PLACES FOR BANQUET'. At the bottom of the form are two radio buttons: 'THIS RESTAURANT MAKE TAKEAWAY' and 'THIS RESTAURANT MAKE DELIVERY'. A blue 'CREATE' button is positioned at the very bottom of the form. The entire form is centered on a light gray background. The top header bar is identical to the one in Figure 8.

Figure 9: Page de création d'un établissement

HOME
Establishment, locality, zip, street, user, email
SEARCH
Hello, hakim
Log out
Admin zone

ESTABLISHMENTS (ACCESSIBLE)






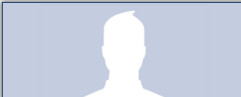


	Chez Théo Sodexho ULB #1 RESTAURANT Avenue Paul Héger n°22, Ixelles 1050 02/650 49 35 http://wwwdev.ulb.ac.be/restaurants/solb... 4 comments
	Mirabelle #2 RESTAURANT Chaussée de Boondael n°459, Ixelles 1050 02/649 51 73 3 comments
	Mano a Mano #3 RESTAURANT Rue Saint-Boniface n°8, Ixelles 1050 02/502 08 01 3 comments
	Le belga queen #4 RESTAURANT Rue Fossé-aux-Loups n°32, Bruxelles 1000 02/217 21 87 http://www.belgaqueen.be/ 6 comments
	Indochine #5 RESTAURANT

Figure 10: Page d'administration des établissements

ADMINISTRATORS

	Boris #1 boris@brusselsbook.be boris@brusselsbook.be wrote 10 comments
	Fred #6 fred@brusselsbook.be fred@brusselsbook.be wrote 18 comments
	hakim #8 hakim@brusselsbook.be hakim@brusselsbook.be wrote 7 comments

USERS

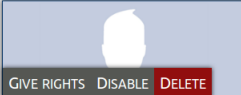
	Ivan #2 ivan@brusselsbook.be ivan@brusselsbook.be wrote 15 comments GIVE RIGHTS DISABLE DELETE
---	---

Figure 11: Page d'administration des utilisateurs

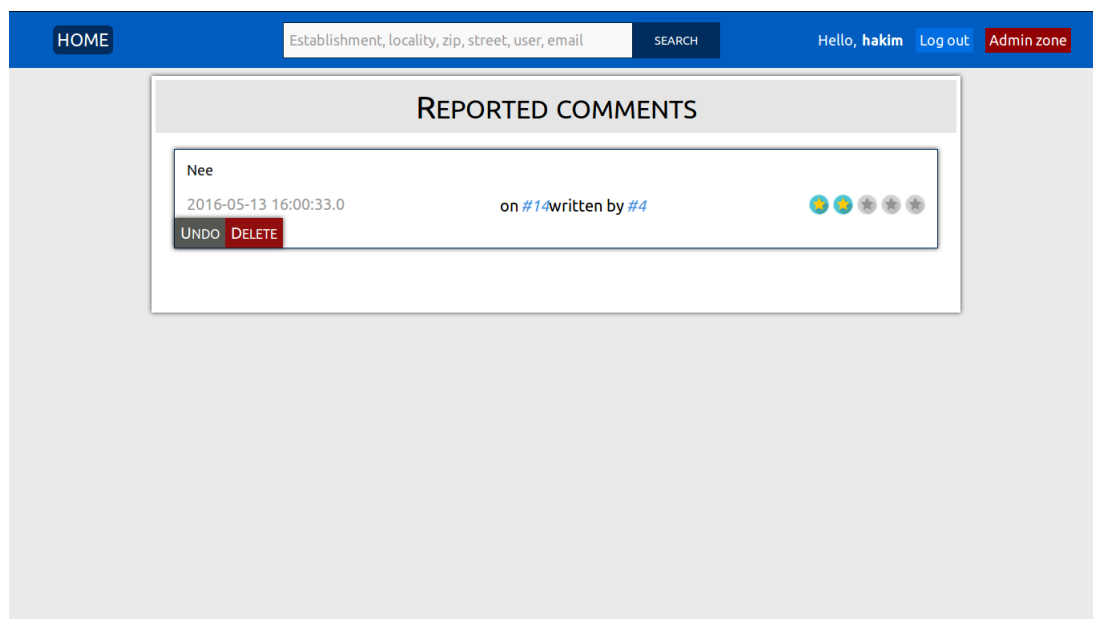


Figure 12: Page de gestion des commentaires signalés