

Ro numberstyle

UNIVERSITÉ LIBRE DE BRUXELLES

Rapport : HORECA

Thomas Perale (408160)

INFO-H-303 Base de données

Esteban Zimányi, Michaël Waumans

Table des matières

1	Diagramme entité association	3
1.1	Diagramme	3
1.2	Contraintes	3
2	Modèle relationnel	3
2.1	Modèle	3
2.2	Contraintes	4
3	Hypothèses	4
4	Justification	4
5	Script SQL DDL	5
6	Requêtes	8
6.1	Tous les Utilisateurs qui apprécient au moins 3 établissements que l'utilisateur <i>Brenda</i> apprécie.	8
6.1.1	Algèbre relationnelle	8
6.1.2	Calcul relationnel	8
6.2	Tous les établissements qu'apprécie au moins un utilisateur qui apprécie tous les établissements que <i>Brenda</i> apprécie.	8
6.2.1	Algèbre relationnelle	9
6.3	Tous les établissements pour lesquels il y a au plus un commentaire.	9
6.3.1	Algèbre relationnelle	9
6.3.2	Calcul relationnel	9
6.4	La liste des administrateurs n'ayant pas commenté tous les établissements qu'ils ont créés.	9
6.4.1	Algèbre relationnelle	10
6.4.2	Calcul relationnel	10
6.5	La liste des établissements ayant au minimum trois commentaires, classée selon la moyenne des scores attribués.	10
6.5.1	Algèbre relationnelle	11
6.5.2	Calcul relationnel	11
6.6	La liste des labels étant appliqués à au moins 5 établissements, classée selon la moyenne des scores des établissements ayant ce label.	11
6.6.1	Algèbre relationnelle	11
6.6.2	Calcul relationnel	11
7	Implémentation	12
7.1	Language et libraires	12
7.2	Script d'insédition de données	12
7.3	Instruction d'installation de l'application	18
7.4	Liaison avec la base de donnée.	18
8	Application	18
8.1	Page principale.	18
8.2	Établissements	18
8.3	Utilisateur	18

9 Apports personnels	18
9.1 Site adapté aux mobiles	18
9.2 Utilisateur administrateur	19
9.3 Localisation des restaurants sur une carte	19
9.4 Géolocalisation sur la carte	20
9.5 Gestion des photos	20
9.6 Recherche de label similaire.	20
9.7 Vote dans les commentaires	21
9.8 Image des restaurants à sont récupérées d'un moteur de recherche	21

1 Diagramme entité association

1.1 Diagramme

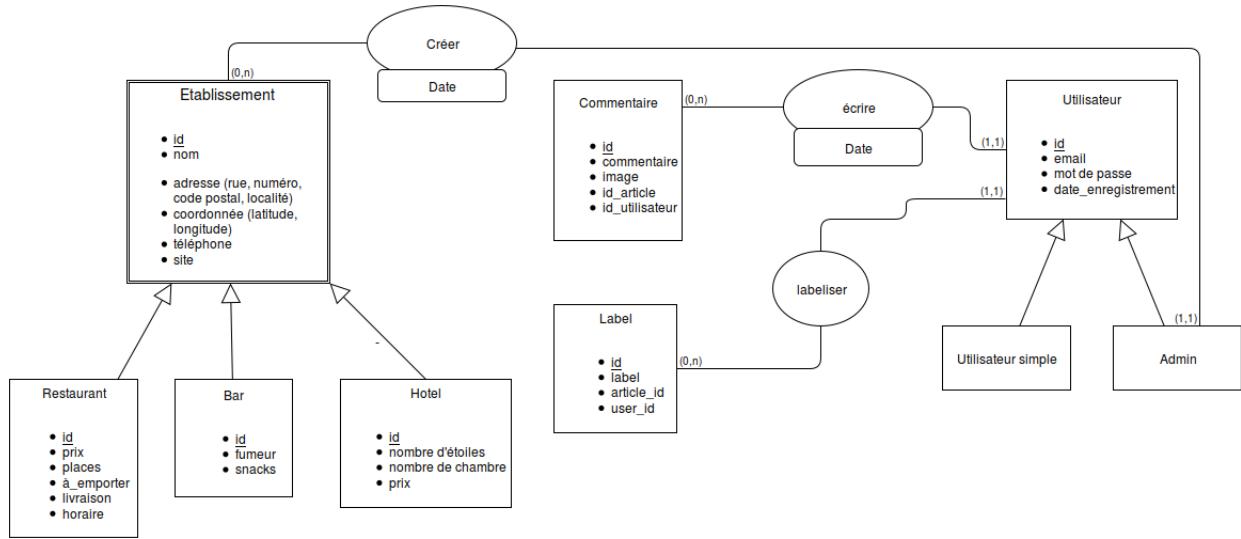


FIGURE 1 – Diagramme entité association

1.2 Contraintes

Les contraintes sont les suivantes :

- Le couple (Longitude, Latitude) est unique.
- La longitude et la latitude doivent être compris entre -180 et 180.
- Le nom d'utilisateur est unique.
- L'email d'utilisateur est unique.
- Pour les commentaires, le couple (IdUtilisateur, date) est unique. Donc un même utilisateur ne peut pas écrire deux commentaires en même temps.
- Pour les labels, le triplet : (Label, IdArticle, IdUtilisateur) est unique, ça veut dire qu'une personne ne peut pas ajouter deux fois le même label.
- Le nombre d'étoile (commentaire et hotel), doit être compris entre 1 et 5.
- Les ID des hotels/bars/restaurants référencent ceux de la table établissement, chaque établissement est soit un hotel, soit un bar, soit un restaurant, il ne peut pas être deux type d'établissement en même temps.
- La Date de création d'un établissement doit précéder celle de ses commentaires.
- La Date d'enregistrement d'un admin doit précéder celle de création d'un établissement par celui-ci.
- La Date d'enregistrement d'un utilisateur doit être antérieur à celle où il a écrit un commentaire.

2 Modèle relationnel

2.1 Modèle

Utilisateur(Id, Email, MotDePasse, DateEnregistrement, Admin(0, 1))

Utilisateur.Admin indique si l'utilisateur est un admin ou non.

Etablissement(Id, Nom, Adresse.Rue, Adresse.Numéro, Adresse.CodePostal, Adresse.Localité, Coordonnée.Latitude, Coordonnée.Longitude, Téléphone, Site, DateCreation, Creator)

Etablissement.Creator référence Utilisateur.Id.

Restaurant(Id, Prix, Places, AEmporter, Livraison, horaire)

Restaurant.Id référence Etablissement.Id.

Bar(Id, Fumeur, Snacks)

Bar.Id référence Etablissement.Id.

Hotel(Id, NombreEtoile, NombreChambre, Prix)

Hotel.Id référence Etablissement.Id.

Commentaire(Id, Commentaire, NombreEtoile, Date, Image, IdArticle, IdUtilisateur)

Commentaire.IdArticle référence Etablissement.Id.

Commentaire.IdUtilisateur référence Utilisateur.Id.

Label(Id, Label, IdArticle, IdUtilisateur)

Commentaire.IdArticle référence Etablissement.Id.

Commentaire.IdUtilisateur référence Utilisateur.Id.

2.2 Contraintes

- On doit d'abord créer l'établissement avant de créer sa spécialisation (restaurant, bar, hotel) de manière à pouvoir référencer cet établissement.

3 Hypothèses

Les utilisateur "admin" sont encodé directement dans la base de donnée.

Il est marqué dans l'énoncé :

Les utilisateurs peuvent commenter plusieurs fois le même établissement à des dates différentes.

Les dates seront stocké dans ma base de donnée sous forme de *timestamp*, dès lors un utilisateur ne pourra pas envoyer deux message lors de la même seconde.

Un administrateur peut modifier un établissement, mais j'ai décidé de ne pas stocker les dates de modification mais seulement celle de création.

Les adresses mails ainsi que les mot de passes des *Utilisateur* qui nous sont fournis dans le fichier *Data.zip* peuvent être choisis.

4 Justification

J'ai décidé pour la généralisation des établissements d'hériter les clés principales qui sont dans établissement. De cette manière j'évite de devoir redéclarer à chaque fois dans chaque table spécialisée (hotel, restaurant ou bar) les mêmes données. De plus la recherche de d'établissement par nom est rendue plus simple.

Pour le généralistion des utilisateurs j'ai décidé d'ajouter une colonne "Admin" dans la table utilisateur pour permettre de savoir si un utilisateur est admin ou non.

En ce qui concerne les *labels* j'ai décidé de mettre l'*ID* de l'utilisateur dans une des colonnes, de cette façon on sait facilement savoir si une personne a déjà ajouté un certain label ou non.

5 Script SQL DDL

..../db/create.sql

```
1 PRAGMA foreign_keys = ON;
2 PRAGMA busy_timeout=30000;
3
4 CREATE TABLE IF NOT EXISTS establishment (
5     id INTEGER PRIMARY KEY AUTOINCREMENT,
6
7     latitude REAL
8         CHECK(-90<=latitude AND latitude <=90) ,
9     longitude REAL
10        CHECK(-180<=longitude AND longitude <=180) ,
11
12     name TEXT NOT NULL,
13
14     address_street TEXT NOT NULL,
15     address_town TEXT NOT NULL,
16     address_number INTEGER NOT NULL,
17     address_zip INTEGER NOT NULL,
18
19     phone_number INTEGER(13) NOT NULL,
20
21     website TEXT,
22         — CHECK (website LIKE '%.%.%'),
23
24     picture BLOB,
25
26     creation_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
27
28     created_by TEXT,
29     FOREIGN KEY (created_by) REFERENCES account (username)
30         ON UPDATE CASCADE
31         ON DELETE CASCADE,
32
33     UNIQUE(latitude , longitude)
34 );
35
36
37 CREATE TABLE IF NOT EXISTS restaurant (
38     id INTEGER PRIMARY KEY NOT NULL REFERENCES establishment(id)
39         ON UPDATE CASCADE
40         ON DELETE CASCADE,
41
42     price INTEGER NOT NULL,
43
44     seat_number INTEGER NOT NULL,
45
46     takeaway BOOLEAN NOT NULL,
47
48     delivery BOOLEAN NOT NULL,
```

```

49      timetable TEXT DEFAULT "0000000000000000"
50  );
51
52
53 CREATE TABLE IF NOT EXISTS bar (
54     id INTEGER PRIMARY KEY NOT NULL REFERENCES establishment(id)
55         ON UPDATE CASCADE
56         ON DELETE CASCADE,
57
58     smokers BOOLEAN,
59
60     snacks BOOLEAN
61 );
62
63 CREATE TABLE IF NOT EXISTS hotel (
64     id INTEGER PRIMARY KEY NOT NULL REFERENCES establishment(id)
65         ON UPDATE CASCADE
66         ON DELETE CASCADE,
67
68     stars INTEGER NOT NULL
69         CHECK(0<=stars AND stars <=5),
70
71     room_number INTEGER NOT NULL
72         CHECK(0<room_number),
73
74     price INTEGER NOT NULL
75 );
76
77 CREATE TABLE IF NOT EXISTS account (
78     username TEXT PRIMARY KEY NOT NULL,
79
80     email TEXT NOT NULL
81         CHECK (email LIKE '%@%.%'),
82
83     password TEXT NOT NULL,
84
85     admin BOOL DEFAULT 0,
86
87     creation_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
88
89     UNIQUE(email)
90 );
91
92 CREATE TABLE IF NOT EXISTS comments (
93     id INTEGER PRIMARY KEY AUTOINCREMENT,
94
95     username TEXT NOT NULL REFERENCES account(username)
96         ON UPDATE CASCADE
97         ON DELETE CASCADE,
98
99     timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
100

```

```

101    establishment_id INTEGER NOT NULL REFERENCES establishment ( id )
102        ON UPDATE CASCADE
103        ON DELETE CASCADE,
104
105    rating INTEGER NOT NULL
106        CHECK(0<rating AND rating <=5),
107
108    comment_text TEXT NOT NULL,
109
110    picture_attached BLOB,
111
112    UNIQUE(username , timestamp, establishment_id)
113 );
114
115 CREATE TABLE IF NOT EXISTS comment_rating (
116     id INTEGER PRIMARY KEY AUTOINCREMENT,
117
118     username TEXT NOT NULL REFERENCES account ( username )
119         ON UPDATE CASCADE
120         ON DELETE CASCADE,
121
122     comments INTEGER NOT NULL REFERENCES comments( id )
123         ON UPDATE CASCADE
124         ON DELETE CASCADE,
125
126     UP BOOLEAN DEFAULT 0 ,
127     DOWN BOOLEAN DEFAULT 0 ,
128
129     timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
130
131     UNIQUE(username , comments)
132 );
133
134 CREATE TABLE IF NOT EXISTS label (
135     id INTEGER PRIMARY KEY AUTOINCREMENT,
136
137     name TEXT NOT NULL,
138
139     username TEXT NOT NULL REFERENCES account ( username )
140         ON UPDATE CASCADE
141         ON DELETE CASCADE,
142
143     establishment_id INTEGER NOT NULL REFERENCES establishment ( id )
144         ON UPDATE CASCADE
145         ON DELETE CASCADE,
146
147     UNIQUE ( name, establishment_id , username )
148 );

```

6 Requêtes

6.1 Tous les Utilisateurs qui apprécient au moins 3 établissements que l'utilisateur *Brenda* apprécie.

Les résultats de cette requête sont visibles dans l'application à la page [/user/Brenda](#).

..../db/1.sql

```
1 —— Tout les utilisateurs qui aiment trois établissement que brenda
2 apprécie .
3 —— Mais qui ne tient pas comptes des personnes qui commentent sur le même
4 —— établissement
5 SELECT DISTINCT c1.username
6 FROM comments AS c1
7     INNER JOIN comments AS c2
8         ON "Brenda"= c2.username
9         AND c1.establishment_id=c2.establishment_id
10         AND c2.rating >=4 AND c1.rating >=4
11 WHERE c1.username!="Brenda"
12 GROUP BY c1.username HAVING COUNT(c1.username)>=3;
```

6.1.1 Algèbre relationnelle

$$\Pi_{c1.username} comments \sigma_{count(c1.username) >= 3} \quad (1)$$

$\bowtie', Brenda' = c2.username \text{ AND } c1.establishment_id = c2.establishment_id \text{ AND } c2.rating >= 4 \text{ AND } c1.rating >= 4$ (2)

6.1.2 Calcul relationnel

$$\{c1.username | comments(c1) \wedge \dots\} \quad (3)$$

$$\exists c2(\text{comments}(c2)) \quad (4)$$

$\wedge c2.username = 'Brenda'$ (5)

$$\wedge c1.establishment_id = c2.establishment_id \quad (6)$$

$$\wedge c1.rating \geq 4 \wedge c2.rating \geq 4 \quad (7)$$

$$\wedge c1.username != 'Brenda' \}) \quad (8)$$

6.2 Tous les établissements qu’apprécie au moins un utilisateur qui apprécie tous les établissements que *Brenda* apprécie.

..../db/2.sql

```
1 -- R2 : Tous les établissements qu'un utilisateur qui
2 -- apprécie tous les établissements que "Brenda" apprécie.
3
4 SELECT DISTINCT e.name
5 FROM establishment AS e, comments AS c
6 WHERE e.id=c.establishment_id
7     AND c.username IN (
```

```

8   SELECT c2.username
9   FROM comments AS c1
10  INNER JOIN comments AS c2
11    ON c1.establishment_id=c2.establishment_id
12    AND c1.username!=c2.username
13    AND c2.rating>=4
14  WHERE c1.username="Brenda"
15    AND c1.rating>=4
16  GROUP BY c2.username HAVING
17    COUNT(c2.username)=COUNT(DISTINCT c1.establishment_id)
18 )
19 ;

```

6.2.1 Algèbre relationnelle

$$\Pi_{e.name} establishment, comments \sigma(e.id AND c.username) \quad (9)$$

$$\exists(\Pi_{c2.username} comments) \quad (10)$$

$$\bowtie_{c1.establishment_id=c2.establishment_id AND c1.username!=c2.username} \quad (11)$$

$$\sigma_{c1.username="Brenda" AND c2.rating>=4} \gamma_{c2.username, COUNT(c2.username)=COUNT(c1.establishment_id)} \quad (12)$$

6.3 Tous les établissements pour lesquels il y a au plus un commentaire.

Les résultats de cette requête sont visible dans l'application à la page `/establishments/discover`

`..../db/3.sql`

```

1 --- Tout les établissement avec maximum un commentaire.
2
3 SELECT c.establishment_id
4 FROM comments c
5 GROUP BY c.establishment_id HAVING COUNT(c.establishment_id)<=1;

```

6.3.1 Algèbre relationnelle

$$\Pi_{c.establishment_id} comments \gamma_{c.establishment_id, count(c.establishment_id)<=1} \quad (13)$$

6.3.2 Calcul relationnel

$$\{c.establishment_id | comments(c) \wedge \quad (14)$$

$$\exists(c.username_1, c.username_2) \quad (15)$$

$$\wedge(c.username_1 \neq c.username_2)\} \quad (16)$$

6.4 La liste des administrateurs n'ayant pas commenté tous les établissements qu'ils ont créés.

..../db/4.sql

```
1 -- Tout les administrateur qui n'ont pas commenté
2 -- tous les établissements qu'ils ont créé
3
4 SELECT a.username
5 FROM account AS a
6 WHERE a.admin=1
7     AND a.username NOT IN (
8         SELECT e.created_by
9         FROM establishment AS e
10        INNER JOIN comments AS c
11            ON e.id = c.establishment_id
12            AND e.created_by != c.username
13        GROUP BY c.username HAVING COUNT(c.establishment_id)>0
14    )
15 ;
```

6.4.1 Algèbre relationnelle

$$\Pi_{a.username} account \quad (17)$$

$$\sigma_{a.admin=1} \text{ AND } \quad (18)$$

$$(\Pi_{e.created_by} establishment) \quad (19)$$

$$\bowtie_{e.id=c.establishment.id} \text{ AND } e.created_by!=c.username \quad (20)$$

$$\gamma_{c.username, \text{COUNT}(c.establishment_id)} > 0 \quad (21)$$

6.4.2 Calcul relationnel

$$\{a.username | account(a) \wedge \quad (22)$$

$$\exists a (a.admin = 1) \wedge \quad (23)$$

$$\neg(e.created_by | establishment(e)) \quad (24)$$

$$\exists c (comments(c) \wedge \quad (25)$$

$$e.id = c.establishment.id \wedge e.created_by != c.username) \quad (26)$$

$$\}) \quad (27)$$

6.5 La liste des établissements ayant au minimum trois commentaires, classée selon la moyenne des scores attribués.

Les résultats de cette requête sont visible dans l'application à la page `/establishments/ratings`

..../db/5.sql

```
1 -- La liste des établissements ayant au minimum trois commentaires,
2 -- classée selon la moyenne des scores attribués.
3
4 SELECT AVG(c.rating), c.establishment_id
5 FROM comments AS c
6 GROUP BY c.establishment_id HAVING COUNT(c.establishment_id)>=3
7 ORDER BY AVG(c.rating);
```

6.5.1 Algèbre relationnelle

$$\Gamma_{AVG(c.rating)} \Pi_{AVG(c.rating), c.establishment_id} comments \sigma_{count(c.establishment_id) >= 3} \Pi_{c.establishment_id} \quad (28)$$

6.5.2 Calcul relationnel

$$\{c.establishment_id | comments(c) \wedge \quad (29)$$

$$\exists c_1, c_2, c_3 (\quad (30)$$

$$c_1.establishment_id = c_2.establishment_id = c_3.establishment_id \quad (31)$$

$$\wedge c_1.username! = c_2.username! = c_3.username) \} \quad (32)$$

6.6 La liste des labels étant appliqués à au moins 5 établissements, classée selon la moyenne des scores des établissements ayant ce label.

Les résultats de cette requête sont visibles dans l'application à la page `/establishments/label/...`

`..../db/6.sql`

```

1 -- La liste des labels étant appliqués à au moins 5 établissements ,
2 -- classée selon la moyenne des scores des établissements ayant ce label .
3
4 SELECT AVG(c.rating) , l.name
5 FROM label AS l
6     JOIN comments AS c
7         ON c.establishment_id=l.establishment_id
8 GROUP BY l.name HAVING COUNT(DISTINCT l.establishment_id) >= 5
9 ORDER BY AVG(c.rating)
10 ;

```

6.6.1 Algèbre relationnelle

$$\Gamma_{AVG(c.rating)} \Pi_{AVG(c.rating), l.name} label \sigma_{count(DISTINCT l.establishment_id) >= 5} \Pi_{l.name} \quad (33)$$

$$\bowtie_{c.establishment_id = l.establishment_id} \quad (34)$$

6.6.2 Calcul relationnel

$$\{l.name | label(l) \wedge \quad (35)$$

$$\exists l_1, l_2, l_3, l_4, l_5 (\quad (36)$$

$$l_1.establishment_id! = l_2.establishment_id! = l_3.establishment_id! = l_4.establishment_id! = l_5.establishment_id \wedge \quad (37)$$

$$l_1.name = l_2.name = l_3.name = l_4.name = l_5.name \quad (38)$$

)}

$$(39)$$

7 Implémentation

7.1 Language et libraires

- *Express.js* comme framework web pour le language *node.js*.
- *Handlebars.js* comme moteur de template pour *express.js*.
- *Bootstrap* et *jQuery* comme framework de frontend.
- *SQLite3* comme moteur de base de donnée.

7.2 Script d'insértion de données

..../init.py

```
1 #!/usr/bin/python
2 # -*- coding: utf-8 -*-
3
4 import dateutil.parser
5 import sqlite3 as lite
6 import sys
7 import datetime
8 from bs4 import BeautifulSoup
9
10 DATA_PATH = "./db/datas/"
11
12 USER_DICT = dict()
13
14 con = lite.connect('./db/test.db')
15
16 def parse_date(date):
17     foo = dateutil.parser.parse(date)
18     print(foo)
19     return foo
20
21 def new_user(username, db, date=None, admin=0):
22     user = USER_DICT.get(username)
23     if user:
24         if date and date < user['date']:
25             cmd = "UPDATE_account SET creation_date=%s WHERE username=%s" % (date, username)
26             print(cmd)
27             db.execute(cmd)
28             user['date'] = date
29     if admin:
30         cmd = "UPDATE_account SET admin=%i WHERE username=%s" % (admin, username)
31         db.execute(cmd)
32     else:
33         if date:
34             # Si l'utilisateur n'existe pas.
35             cmd = "INSERT INTO account (username, email, password, admin, creation_date) VALUES ('%s', '%s', 'default', '%i', '%s')" %
36             (username, username + "@gmail.com", admin, date)
37             print(cmd)
```

```

37         db . execute (cmd)
38     else :
39         cmd = "INSERT INTO account (username , email , password , admin) "
40         VALUES ('%s' , '%s' , ' default ' , %i)" % (username , username +
41             "@gmail . com" , admin)
42         print (cmd)
43         db . execute (cmd)
44         date = datetime . datetime . now ()
45
46
47         USER_DICT [username] = dict ()
48         USER_DICT [username] [ 'date'] = date
49
50
51     con . commit ()
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83

```

```

84     db.close()
85
86     obj = {}
87     info = cafe.find('informations')
88     name = info.find('name').text # Bar name.
89     address_street = info.find('address').find('street').text
90     address_number = info.find('address').find('num').text
91     address_zip = info.find('address').find('zip').text
92     address_town = info.find('address').find('city').text
93     longitude = info.find('address').find('longitude').text
94     latitude = info.find('address').find('latitude').text
95
96     phone_number = info.find('tel').text
97     website = info.find('website')
98     if website:
99         website = website.text
100
101    cmd = "insert_into_establishment_"
102    .....values_(%f,%f,'%s','%s','%s','%s,%i','%s','%s','%s','%s')%" %
103    \
104    .....(float(latitude), float(longitude), name, address_street,
105    .....address_town, address_number, int(address_zip),
106    .....phone_number, website, created_by, creation_date)
107    print(cmd)
108
109    db = con.cursor()
110    db.execute("insert_into_establishment_"
111    .....(latitude,longitude,name,address_street,address_town,address_number,address_z
112    .....values(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)" ,
113    .....(float(latitude), float(longitude), name, address_street,
114    .....address_town, address_number,
115    .....int(address_zip), phone_number, website,
116    .....creation_date, created_by))
117    con.commit()
118    db.close()
119
120    _id = db.lastrowid
121
122    smokers = bool(info.find('smoking'))
123    snacks = bool(info.find('snack'))
124
125    cmd = "INSERT_INTO_bar_(id ,smokers ,snacks )_VALUES_(%i,%i,%i)" %
126    .....(_id ,smokers , snacks)
127    db = con.cursor()
128    db.execute(cmd)
129    con.commit()
130    db.close()
131
132    db = con.cursor()
133    parse_comments(cafe.findall('comment'), _id , db)
134    db.close()

```

```

128
129     db = con.cursor()
130     parse_label(cafe.find_all('tag'), _id, db)
131     db.close()
132
133 def parse_restaurant(s, conn):
134     """
135     Parse bar file.
136     """
137     for cafe in s.find_all("restaurant"):
138         created_by = cafe['nickname']
139         creation_date = parse_date(cafe['creationdate'])
140
141         db = con.cursor()
142         new_user(created_by, db, creation_date, 1)
143         db.close()
144
145         obj = {}
146         info = cafe.find('informations')
147         name = info.find('name').text # Bar name.
148         address_street = info.find('address').find('street').text
149         address_number = info.find('address').find('num').text
150         address_zip = info.find('address').find('zip').text
151         address_town = info.find('address').find('city').text
152         longitude = info.find('address').find('longitude').text
153         latitude = info.find('address').find('latitude').text
154
155         phone_number = info.find('tel').text
156         website = info.find('website')
157         if website:
158             website = website.text
159
160         cmd = "insert_into_establishment_"
161         .....values_(%f,%f,'%s','%s','%s',%s,%i,'%s','%s','%s','%s')%" %
162             \
163             (float(latitude), float(longitude), name, address_street,
164              address_town, address_number, int(address_zip),
165              phone_number, website, created_by, creation_date)
166         print(cmd)
167
168         db = con.cursor()
169         db.execute("insert_into_establishment_"
170             (latitude,longitude,name,address_street,address_town,address_number,address_z
171             .....values(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)",
172             (float(latitude), float(longitude), name, address_street,
173              address_town, address_number,
174              int(address_zip), phone_number, website,
175              creation_date, created_by))
176         con.commit()
177         db.close()
178

```

```

173     _id = db.lastrowid
174
175     c = info.find('closed')
176     closed = [0 for i in range(14)]
177     if c:
178         for i in c.findall('on'):
179             day = int(i['day'])
180             print("Day %i" % day)
181             if i.get('hour'):
182                 if i['hour'] == 'am':
183                     closed[day * 2] = 1
184                 elif i['hour'] == 'pm':
185                     closed[(day * 2) + 1] = 1
186             else:
187                 closed[day * 2] = 1
188                 closed[(day * 2) + 1] = 1
189
190     timetable = ""
191     for i in closed:
192         timetable = timetable + str(i)
193
194     print("----->TIMET" + timetable)
195
196     takeaway = bool(info.find('takeaway'))
197     delivery = bool(info.find('delivery'))
198     price = int(info.find('pricerange').text)
199     seat_number = int(info.find('banquet')['capacity'])
200
201     db = con.cursor()
202     db.execute("INSERT INTO restaurant"
203                 "(id, timetable, takeaway, delivery, price, seat_number) VALUES"
204                 "(?, ?, ?, ?, ?, ?)",
205                 [_id, timetable, takeaway, delivery, price, seat_number])
206     con.commit()
207     db.close()
208
209     db = con.cursor()
210     parse_comments(cafe.findall('comment'), _id, db)
211     db.close()
212
213
214
215
216
217 if __name__ == "__main__":
218     parse_bar(BeautifulSoup(open(DATA_PATH + "Cafes.xml").read(),
219                             'html.parser'), con)
220     parse_restaurant(BeautifulSoup(open(DATA_PATH +
221                                     "Restaurants.xml").read(), 'html.parser'), con)

```


7.3 Instruction d'installation de l'application

À la racine du projet.

```
1 make  
2 npm install  
3 node index.js
```

7.4 Liaison avec la base de donnée.

L'accès par l'application à la base de donnée se fait par l'ensemble des fonctions stockées dans le dossier `./db/`.

`database_utils.js` Ensemble de fonctions qui interagissent avec tout les établissements.

`restaurant_db_utils.js` Ensemble de fonctions qui interagissent avec les restaurants.

`bar_db_utils.js` Ensemble de fonctions qui interagissent avec les bars.

`hotel_db_utils.js` Ensemble de fonctions qui interagissent avec les hotels.

`label_db_utils.js` Ensemble de fonctions qui interagissent avec les labels.

`comment_utils.js` Ensemble de fonctions qui interagissent avec les commentaires.

8 Application

8.1 Page principale.

La page principale est la première page de l'application que l'utilisateur va voir. Dans cette page, plusieurs chose sont déjà possible :

- S'inscrire
- Se connecter
- Choisir parmis une selection d'établissement pris au hasard.
- Accéder à une selection d'établissement plus spécifique.
- Une carte des établissements qui sont sur le site.

8.2 Établissements

La page des établissements donne à l'utilisateur un coup d'œil sur ses caractéristiques propres. Mais c'est aussi un lieu pour les utilisateurs de donner leur avis sur l'établissement et de le classer dans des catégories. Chaque commentaire qu'un utilisateur laisse peut être mis en avant par un système de vote individuel au commentaire, de cette façon les commentaires qui sont malhonnêtes selon les utilisateurs sont visibles.

8.3 Utilisateur

Chaque utilisateur a droit à une petite page personnelle qui récapitule les quelques donnée personnelle ainsi que les interactions que cet utilisateur a eu avec l'application, comme par exemple les commentaires ou les labels utilisés.

9 Apports personnels

9.1 Site adapté aux mobiles

Pour que l'application soit adaptée au type de consommation actuelle, elle a été créée dans l'idée de pouvoir aussi bien être facile d'utilisation sur ordinateur que sur mobile.

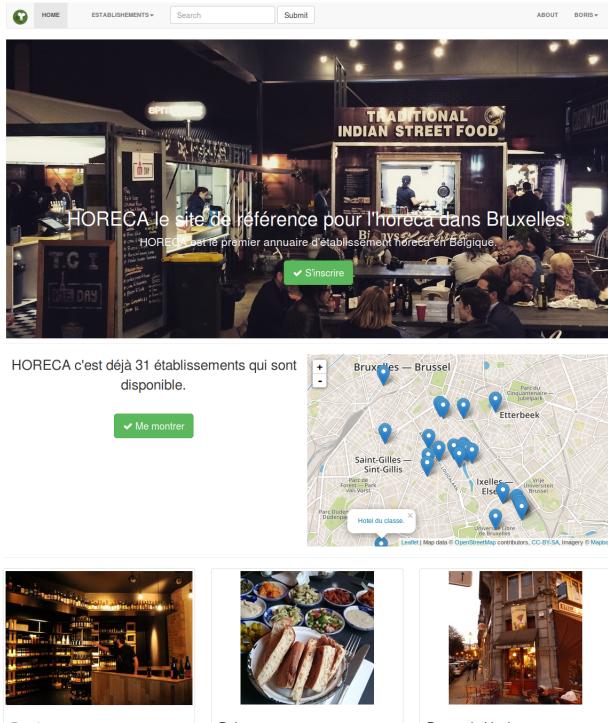


FIGURE 2 – Page principale.

9.2 Utilisateur administrateur

Des comptes administrateur sont à disposition pour pouvoir gérer les publications, si celle-ci doivent être changé, ou pour faire un travail de modération dans les commentaires ou les labels. Un administrateur a le pouvoir de :

- Supprimer un commentaire.
- Supprimer un label.
- Supprimer un établissement.
- Supprimer un utilisateur.
- Ajouter d'autres utilisateur au poste d'administrateur.
- Crée des restaurants.
- Crée des hotels.
- Crée des bars.
- Modifier les données des établissements.

9.3 Localisation des restaurants sur une carte

Afin de permettre de pouvoir directement voir l'emplacement des établissements sur une l'application intègre des cartes pour visualiser l'emplacement de l'établissement *HORECA*. Celles-ci sont intégrées à la page web à l'aide d'une bibliothèque Javascript, Leaflet. En plus de montrer l'emplacement sur une carte celles-ci montre aussi un rapide aperçu des caractéristiques de l'établissement.

9.4 Géolocalisation sur la carte

En plus de montrer la localisation des restaurants, l'application permet de pouvoir se géolocaliser sur la carte, par rapport aux établissements.

Cette fonctionnalité est d'autant plus la bienvenue que l'application est adaptée aux téléphones mobiles.

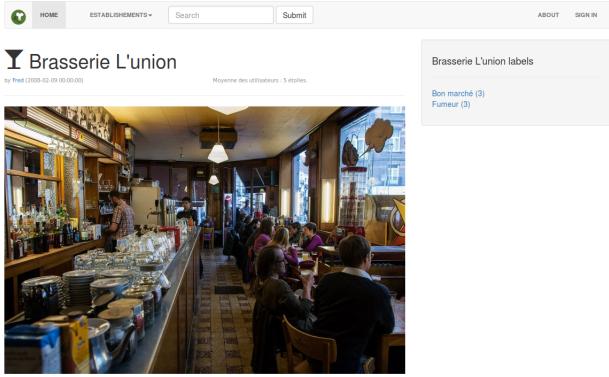


FIGURE 3 – Page principale.

This screenshot shows a user profile page for 'Admin : Boris'. At the top, it displays the email 'Boris@gmail.com' and the creation date 'Crée le 2003-04-04 00:00:00'. Below this, there are two sections: 'Commentaires' and 'Labels'. The 'Commentaires' section lists four comments with their respective dates and text snippets. The 'Labels' section lists three labels with their descriptions. The URL at the bottom of the page is 'http://localhost:3000/user/Boris<-11/11/11'.

Commentaires	Labels
Commentaire du 2008-01-06 00:00:00 Très plein les soirs de match, bruyant et enflamé.	Label Fumeur
Commentaire du 2008-02-04 00:00:00 Sympa à l'extérieur en été mais très bruyant à l'intérieur.	Label Bon marché
Commentaire du 2008-02-05 00:00:00 Bonne ambiance à partir de 23h. Déco apaisante. Spécialité : sous marin.	Label Terrasse

FIGURE 4 – Page utilisateur.

9.5 Gestion des photos

L’application gère le stockage d’image dans la base de donnée. Chaque utilisateur inscrit sur le site peuvent dès lors écrire un commentaire dans la section adaptée accompagné d’une photo de l’établissement visité par exemple.

Quant aux utilisateurs dit *administrateur*, ils peuvent modifier la photo de l’établissement pour que chaque visiteur ait un aperçu de l’apparence de l’établissement.

De plus la gestion de ces photos permet de les visionner de manière confortable à l’aide du module enhance.js. En effet celui-ci permet de voir une photo *en grand*, sans devoir changer de page.

9.6 Recherche de label similaire.

Pour permettre aux visiteurs de trouver des établissements avec des contraintes spécifiques, l’application lie tout les labels de chaque établissements. Un visiteur peut dès lors retrouver tout les établissements qui possède le label de son choix, comme par exemple *Fumeur*.

9.7 Vote dans les commentaires

Un système de vote individuel à chaque commentaire est mis en place pour que chaque utilisateur inscrit ait le droit de juger des commentaires qui selon lui ne sont pas raisonnable ou inversément.

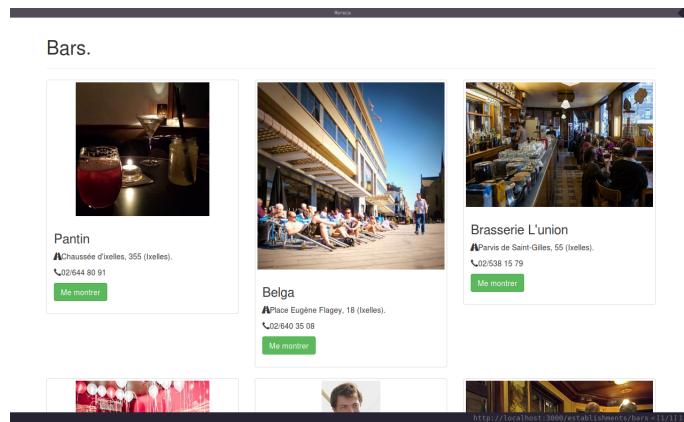


FIGURE 5 – Site responsive.



FIGURE 6 – Établissement indiqué sur une carte.

9.8 Image des restaurants à sont récupérées d'un moteur de recherche

Les images qui représente les restaurants sont récupérées du moteur de recherche libre *Searx*. C'est implémenté pour donner plus de diversité dans le site. L'application aurait pu simplement affiché une image par défaut mais cette solution est plus marrante bien que certains résultats ne soient pas toujours cohérents.

Chez Théo Sodexho ULB

by Boris (2008-02-10 00:00:00)

Moyenne des utilisateurs : 2.5 étoiles.

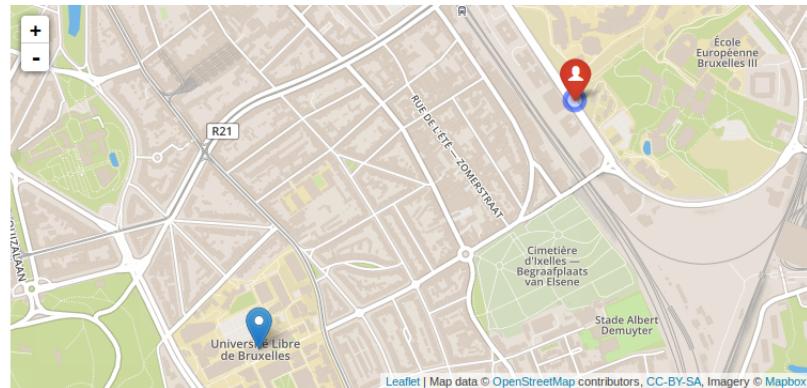
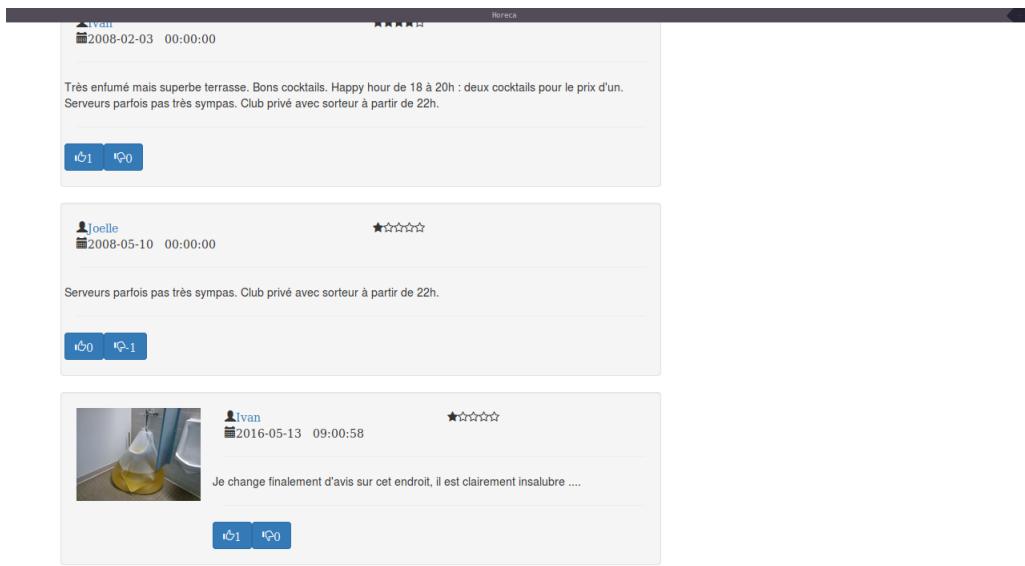


FIGURE 7 – Un marqueur indique l'endroit où l'on se trouve.



ivan 2008-02-03 00:00:00 Horreca

Très enfoncé mais superbe terrasse. Bons cocktails. Happy hour de 18 à 20h : deux cocktails pour le prix d'un. Serveurs parfois pas très sympas. Club privé avec sorteur à partir de 22h.

joelle 2008-05-10 00:00:00

Serveurs parfois pas très sympas. Club privé avec sorteur à partir de 22h.

ivan 2016-05-13 09:00:58

Je change finalement d'avis sur cet endroit, il est clairement insalubre



[http://localhost:3000/establishments/bar/2-\(1/1\).bat](http://localhost:3000/establishments/bar/2-(1/1).bat)

FIGURE 8 – Ici un commentaire accompagné d'une photo.

Horera

HOME ESTABLISSEMENTS Search Submit ABOUT IVAN

Il y a 4 établissements qui utilisent le label "Bar à vin"



Marché au vin
Rue du Belvédère, 14 (Ixelles).
02/640 56 10
[Me montrer](#)



Delecta
Rue Lannoy, 2 (Ixelles).
02/644 19 49
[Me montrer](#)



Picolo
[http://localhost:3000/label/Bar à vin-\[1/1\].top](http://localhost:3000/label/Bar à vin-[1/1].top)

FIGURE 9 – Page regroupant les établissements sous le label "Bar à vin".

Horera

ivan 2008-02-03 00:00:00 ★★★★

Très enfoncé mais superbe terrasse. Bons cocktails. Happy hour de 18 à 20h : deux cocktails pour le prix d'un. Serveurs parfois pas très sympas. Club privé avec sorteur à partir de 22h.

[1/1](#) [1/0](#)

joelle 2008-05-10 00:00:00 ★★★★★

Serveurs parfois pas très sympas. Club privé avec sorteur à partir de 22h.

[1/0](#) [1/1](#)

ivan 2016-05-13 09:00:58 ★★★★★

Je change finalement d'avis sur cet endroit, il est clairement insalubre

[1/1](#) [1/0](#)

[http://localhost:3000/establishments/bar/2-\[1/1\].top](http://localhost:3000/establishments/bar/2-[1/1].top)

FIGURE 10 – Des votes en dessous de chaque commentaire.

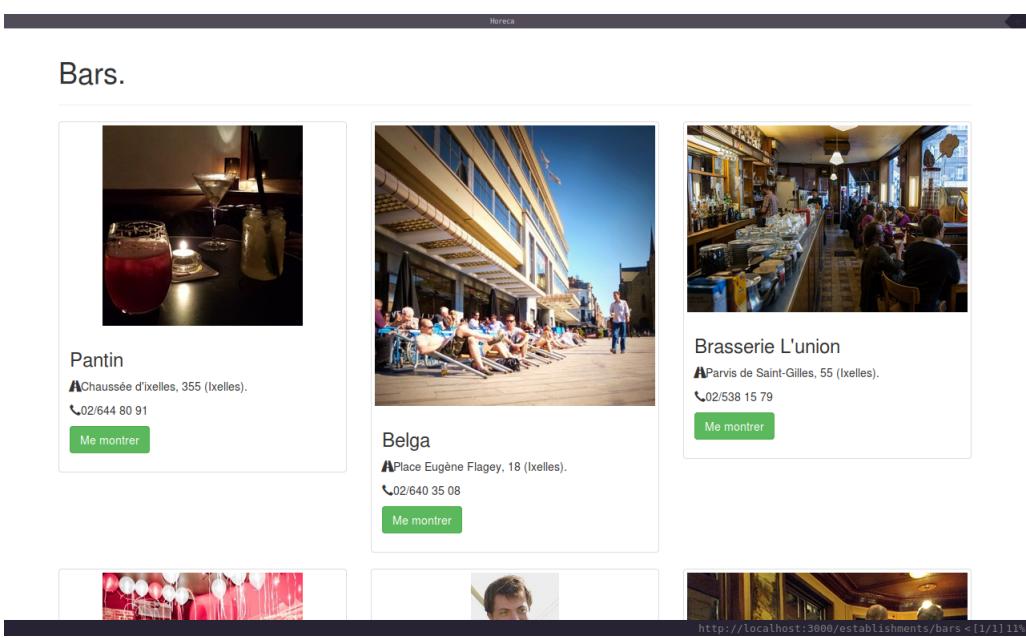


FIGURE 11 – Image tirée d'un moteur de recherche.