

# **Rapport du projet *Enjoy!***

## *Projet de base de données*

Réalisé par :

DUCOUDRÉ Max

GUILLET Elsa

GIBOZ Alexandre

## Table des matières

<b>Présentation du projet :</b> .....	3
<b>Entité-association :</b> .....	4
<b>Diagramme d'entité-association</b> .....	4
<b>Éclaircissements :</b> .....	4
<b>Limites du diagramme :</b> .....	4
<b>Modèle relationnel :</b> .....	5
<b>Tables :</b> .....	5
<b>Clés étrangères :</b> .....	5
<b>Contraintes :</b> .....	6
<b>Maquette du site WEB</b> .....	7
<b>Connexion utilisateur :</b> .....	7
<b>Inscription Utilisateur :</b> .....	8
<b>Tableau de bord utilisateur</b> .....	9
<b>Tableau de bord livreur</b> .....	10
<b>Recherche restaurant</b> .....	12

## Présentation du projet :

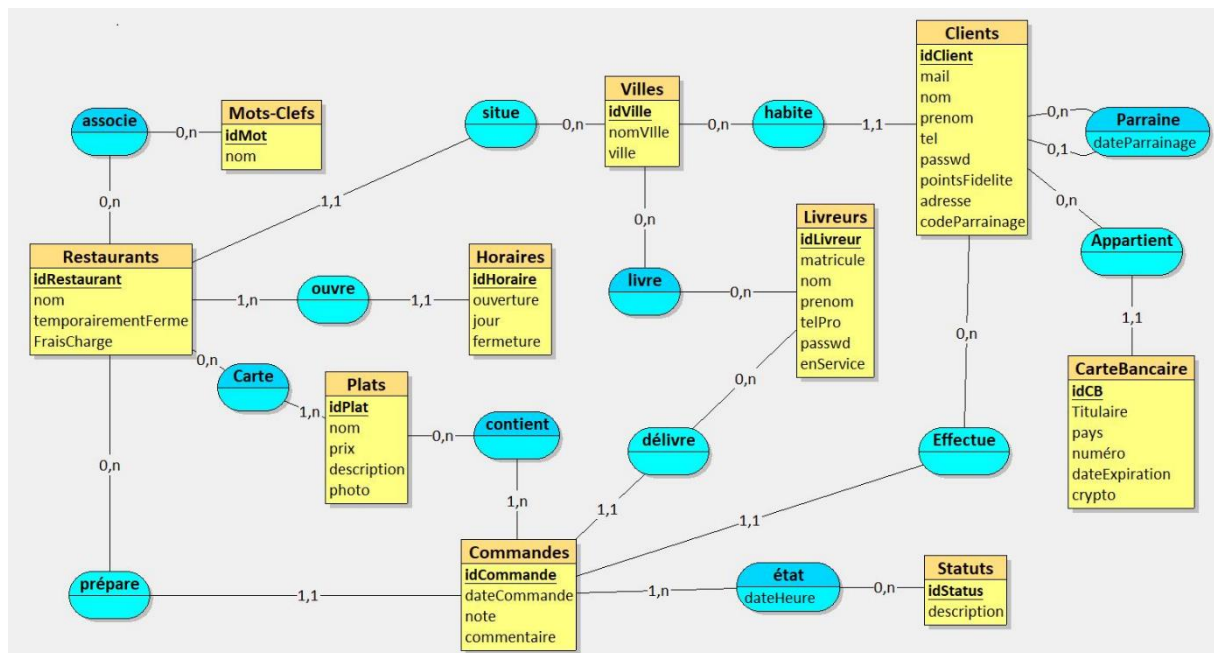
« Enjoy ! » est une plateforme de livraison de plats à domicile dont le but est d'offrir à ses utilisateurs un vaste choix de restaurants, tout en permettant aux restaurants partenaires de ne pas avoir à gérer leur propre service de livraison.

Ainsi, Enjoy ! ne vend pas directement de nourriture, mais se contente de transmettre aux restaurants les commandes passées sur la plateforme, et met ensuite à disposition des restaurants sa flotte de livreurs afin d'acheminer les plats préparés jusqu'au domicile des clients.

*Plus de détails sur « consignes\_enjoy.pdf »*

## Entité-association :

### Schéma d'entité-association



mcd.jpg

### Éclaircissements :

Un « horaire » représente une « tranche » de journée d'ouverture pour un restaurant.

Un « statuts » représente l'état d'une commande, il peut être : « en attente », « en préparation », « en cours de livraison », « livré » et « annulé ».

L'attribut « codeParrainage » est un code unique que le client peut partager à ses futurs filleuls.

### Limites du Schéma :

À partir de ce diagramme, on peut décèler la structure d'une base de données relationnelle.

Néanmoins, ce diagramme ne permet pas de satisfaire l'intégralité des contraintes mises en lumière dans le cahier des charges, qu'elles soient explicites ou non. Les cas suivants devront être traités en aval :

- Un restaurant peut avoir un horaire d'ouverture qui déborde sur un horaire de fermeture.
- Une commande peut avoir un avis et une note alors que son statut n'est pas défini sur « livré ».
- Un client peut effectuer une commande sans avoir de carte bancaire au préalable.
- Un livreur peut avoir une commande qui n'appartient pas à sa ville.
- Les dates de changement de statuts (état) peuvent ne pas être dans le bon ordre chronologique.
- Un livreur peut avoir une commande à sa charge sans être en service.
- Si un restaurant est « temporairement fermé » ou qu'il est fermé due à ses « horaires », il peut avoir des commandes.

Néanmoins, l'intégralité des limites mentionnées peuvent être corrigées au travers de triggers et d'une implémentation fiable de règles dans la partie « backend » de l'application.

Note : Ne sont pas listés ici les soucis pouvant être corrigés avec des « constraints » et des « checks » lors de l'implémentation de ce schéma SQL.

## Modèle relationnel :

Il est possible de traduire ce schéma (aussi appelé « Modèle Conceptuel de Données ») en un « Modèle Relationnel », qui va décrire la structure de la base de données avec une approche plus concrète et proche de la notion mathématique des ensembles et des relations.

### Tables :

Mots\_Clefs = (idMot SERIAL, nom VARCHAR(50));

Plats = (idPlat SERIAL, nom VARCHAR(50), prix DECIMAL(15,2), description VARCHAR(50), photo VARCHAR(50));

Livreurs = (idLivreur SERIAL, matricule CHAR(8), nom VARCHAR(50), prenom VARCHAR(50), telPro VARCHAR(50), passwd VARCHAR, enService BOOL);

Villes = (idVille SERIAL, nomVille, ville CHAR(5));

Clients = (mail VARCHAR(50), idClient INT, #parrain INT, nom VARCHAR(50), prenom VARCHAR(50), tel VARCHAR(50), passwd VARCHAR(50), pointsFidelite INT, adresse VARCHAR(50), #idVille, dateInscription DATE);

Statuts = (idStatus SERIAL, description VARCHAR(50));

Restaurants = (idRestaurant SERIAL, nom VARCHAR(50), temporairementFerme LOGICAL, FraisCharge DECIMAL(15,2), #idVille);

Horaires = (idHoraire SERIAL, ouverture TIME, jour INT, fermeture TIME, #idRestaurant);

Commandes = (idCommande SERIAL, dateCommande DATE, note INT, commentaire VARCHAR(50), #idRestaurant, #idLivreur, #mail);

Carte = (#idRestaurant, #idPlat);

associe = (#idRestaurant, #idMot);

livre = (#idLivreur, #idVille);

contient = (#idPlat, #idCommande);

état = (#idCommande, #idStatus, dateHeure DATETIME);

### Clés étrangères :

FK : Restaurants(idVille) référence Villes(idVille)

FK : Clients(idVille) références Villes(idVille)

FK : Horaires(idRestaurant) référence Restaurants(idRestaurant)

FK : Commandes(idRestaurant) référence Restaurants(idRestaurant)

FK : Commandes(idLivreur) référence Livreurs(idLivreur)

FK : Commandes(mail) référence Client(mail)

FK : Carte(idRestaurant) référence Restaurants(idRestaurant)

FK : Carte(idPlat) référence Plats(idPlat)

FK : associe (idRestaurant) référence Restaurants(idRestaurant)

FK : associe (idMot) référence Mots\_clefs(idMot)

FK : livre(idLivreur) référence Livreurs(idLivreur)  
FK : livre(idVille) référence Ville(idVille)  
FK : contient(idPlat) référence Plats(idPlat)  
FK : contient (idCommande) référence Commandes(idCommande)  
FK : état(idCommande) référence Commandes(idCommande)  
FK : état(idStatus) référence Statuts(idStatus)  
FK : clients(parrain) référence Clients(mail)

## Contraintes :

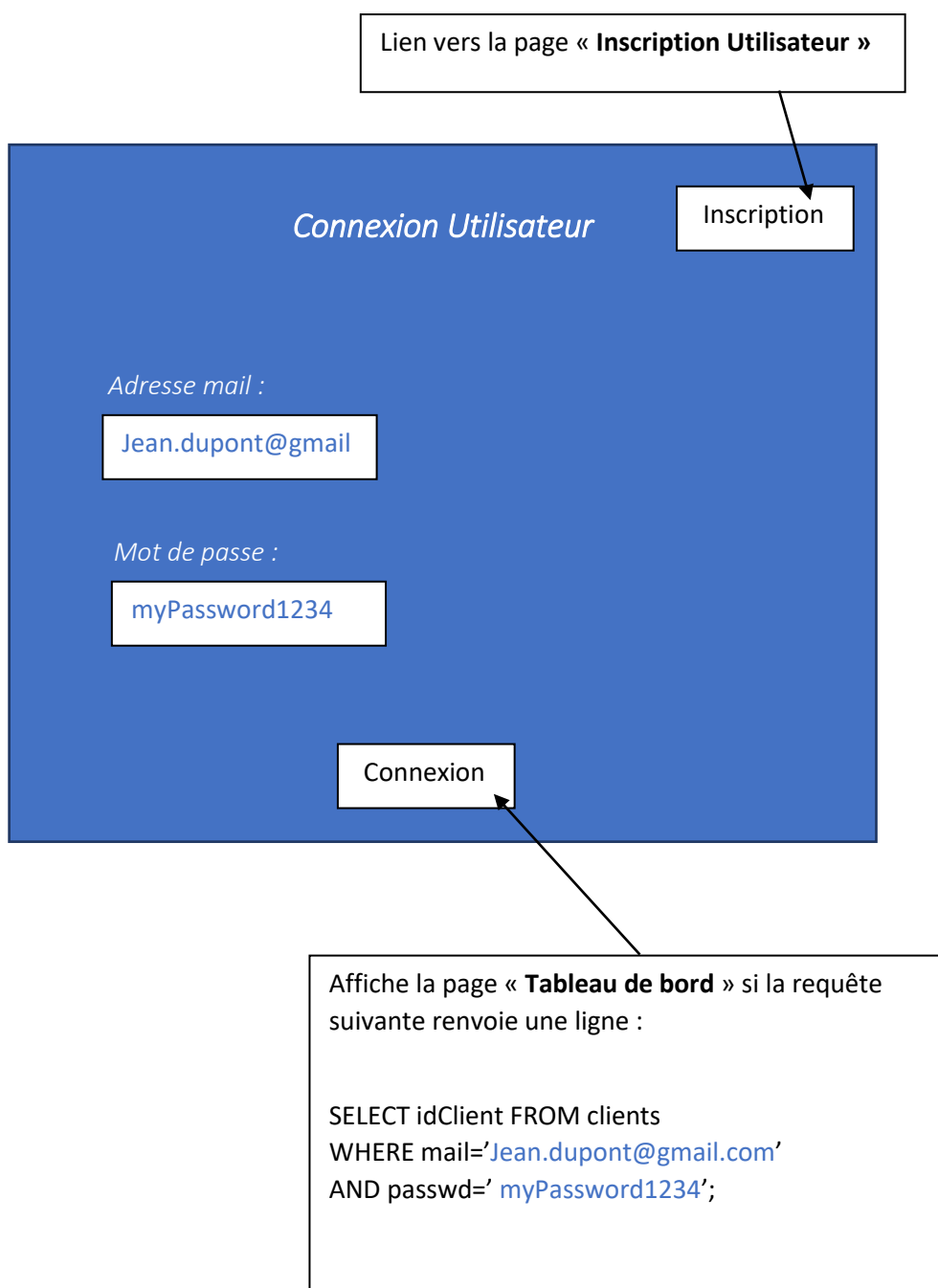
Afin de respecter au mieux le cahier des charges, certains attributs doivent se voir attribuer des contraintes, et ce pour l'incohérence des données stockées et manipulées ultérieurement par l'application :

- Dans la table « **commandes** » :
  - Une note doit être comprise entre 0 et 5.
  - Une note et un commentaire peuvent être NULL si le client n'a pas laissé d'avis sur la commande.
- Dans la table « **plats** »
  - Un plat a forcément un nom (attribut « nom » en NOT NULL).
  - Le prix d'un plat ne peut être inférieur à 0.
- Dans la table « **clients** »
  - Les points de fidélité sont définis à 0 par défaut, et ces derniers ne peuvent être inférieurs à 0.
- Dans la table « **carte\_bancaire** »
  - Chaque attribut de la carte aura pour type une chaîne de caractère de longueur 50 fixe. En effet, chaque donnée sera chiffrée afin de garantir la confidentialité d'une donnée comme défini par la CNIL..
- Dans la table « **horaires** »
  - Un jour doit être un entier entre 0 et 6, afin de correspondre aux 7 jours de la semaine.
  - L'ouverture ne peut être inférieure à la fermeture (pour des raisons de cohérence).
- Dans la table « **restaurants** »
  - L'attribut « fraisCharge » ne peut être inférieur à 0 car un frais ne peut en aucun cas être négatif.

Lors de la création des tables, les contraintes suivantes seront spécifiées en SQL : Le fichier « *dump.sql* » contient les requêtes « CREATE TABLE » pour créer les tables de la base de données et contient des requêtes « INSERT » pour donner un jeu de données d'exemple.

## Maquette du site WEB

Connexion utilisateur :



## Inscription Utilisateur :

### Inscription utilisateur

Connexion

Adresse mail :  
jean.dupont@gmail.com

Code parrain :  
1BCD

Lien vers la page « Connexion utilisateur »

Nom :  
Dupont

Prénom :  
Jean

Téléphone :  
06 01 02 03 04

Mot de passe :  
myPassword

Confirmation :  
myPassword

Adresse  
10 rue de la paix

Code postal  
45200

Ville :  
Montargis

Liste déroulante des villes obtenue grâce à la requête suivante :  
**SELECT DISTINCT** nomVille, idVille  
**FROM** villes  
**WHERE** codePostal = '45200';

Inscription

Affiche la page « **Tableau de bord utilisateur** » si la requête suivante ne retourne pas d'erreur :

```
INSERT INTO clients(mail, nom, prenom, tel, passwd, adresse, idVille, parrain)  
VALUES ('jean.dupont@gmail.com', 'Dupont', 'Jean', '0601020304', 'myPassword',  
'10 rue de la paix', 9, 2);
```

2 correspond à l'idClient rattaché au code de parrainage '1BCD' récupéré via la requête suivante :

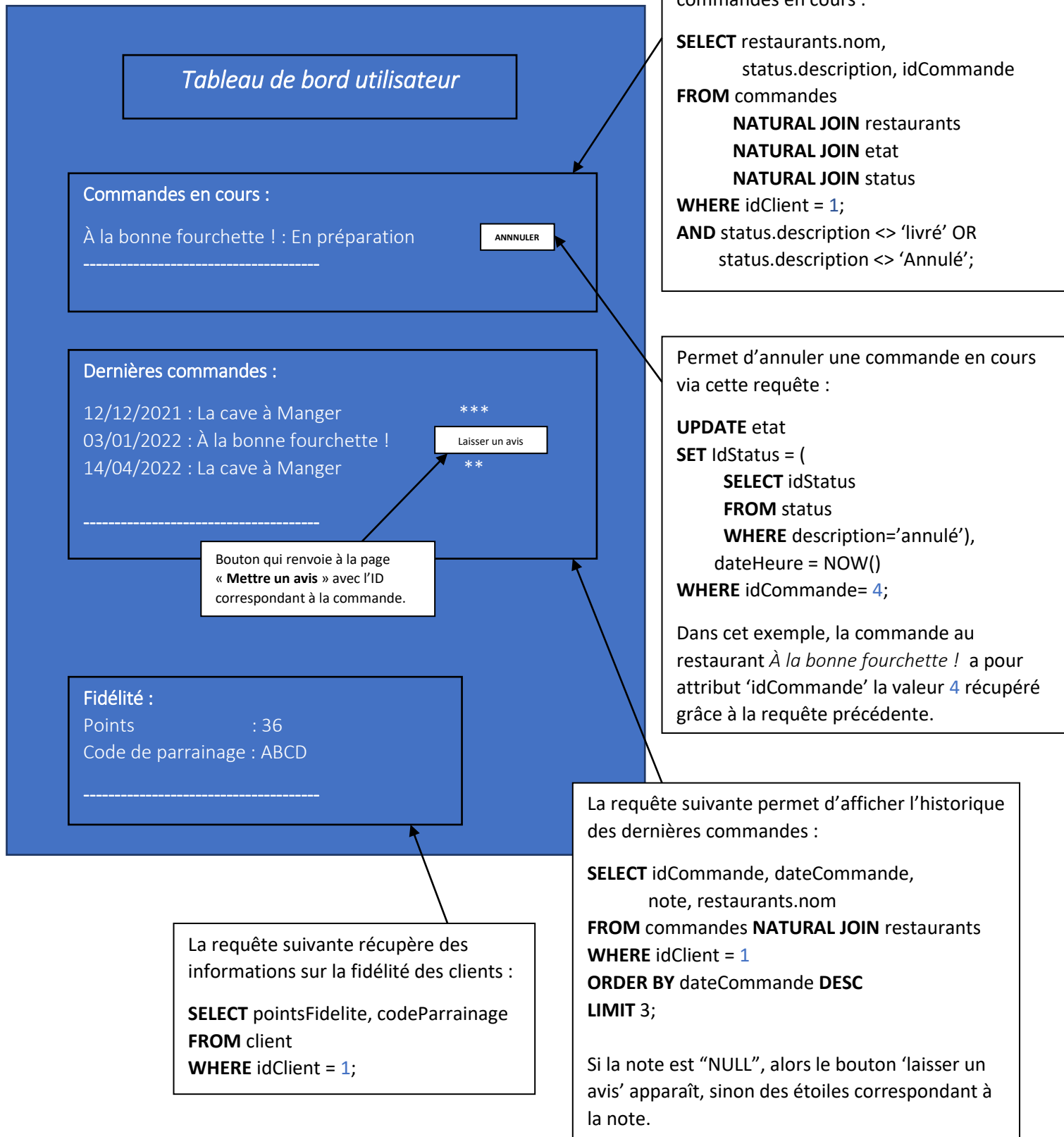
```
SELECT idClient FROM clients WHERE codeParrainage = '1BCD' ;
```

9, récupéré via la requête précédente, correspond à la ville de Montargis.



## Tableau de bord utilisateur

On suppose connaître l'attribut 'idClient' de la table « client » correspondant au client actuellement connecté. (Cet 'idClient' est égal à 1 dans notre exemple).



## Tableau de bord livreur

On suppose connaître l'attribut 'idLivreur' de la table « livreur » correspondant au livreur actuellement connecté. (Cet 'idLivreur' est égal à **3** dans notre exemple).

### Tableau de bord livreur

#### Commandes en attente :

De : 10 rue des Lauriers, Paris 15 (À la bonne fourchette !)  
Vers : 15 place de la République, Paris 16 (Jean Dupont)

PRENDRE LA COMMANDE

De : 10 rue des trèfles, Lyon (La cave à Manger)  
Vers : 15 avenue de la paix, Lyon (Gérard Depardieu)

PRENDRE LA COMMANDE

Les requêtes suivantes se déclenchent lorsque le livreur prend une commande en attente :

-- Mise à jour du statut

```
UPDATE etat
SET idStatus=(
    SELECT idStatus
    FROM status
    WHERE description='en cours de livraison')
AND date=NOW()
WHERE idCommande=5;
```

-- Mise à jour de la commande

```
UPDATE commandes
SET idLivreur = 3
WHERE idCommande = 5;
```

Dans notre exemple, **5** correspond à l'idCommande de la commande choisie par le livreur **3**.  
L'idClient **1** correspond à celui de la commande **5** récupéré avec la requête précédente.

La requête ci-dessous permet de récupérer les commandes en attente que le livreur peut récupérer. C'est-à-dire que le livreur couvre la ville du client et du restaurant.

```
SELECT idCommande,
       client.prenom|'|client.nom as client
       clients.adresse as clientAdresse
       clients.idVille as clientVille
       restaurants.nom
       restaurants.adresse as restaurantAdresse
       restaurants.idVille as restaurantVille

FROM commandes
NATURAL JOIN clients
NATURAL JOIN livre
NATURAL JOIN livreur
NATURAL JOIN etat
NATURAL JOIN ville
NATURAL JOIN restaurant

WHERE livreur.idLivreur = 3
      AND etat.idStatus = (
          SELECT idStatus
          FROM status
          WHERE description='en préparation'
              OR description='en attente')
      AND clients.idVille IN (
          SELECT idVille
          FROM ville NATURAL JOIN livre
          WHERE idLivreur = 3 )
      AND restaurants.idVille IN (
          SELECT idVille
          FROM ville NATURAL JOIN livre
          WHERE idLivreur = 3 );
AND commande.idLivreur IS NULL;
```

## Tableau de bord livreur (suite)

### Villes desservies :

Paris 16

SUPPRIMER

Paris 15

SUPPRIMER

Lyon

SUPPRIMER

Montargis

AJOUTER

Pour qu'un livreur supprime une ville qui ne dessert plus :

```
DELETE FROM livre WHERE idVille = 12  
AND idLivreur = 3;
```

L'idVille 12 correspond à Lyon

La requête suivante permet de récupérer toutes les villes que dessert un livreur :

```
SELECT nomVille, idVille  
FROM villes NATURAL JOIN livre  
WHERE idLivreur = 3 ;
```

Correspond à un menu déroulant affichant la liste des villes récupérées via cette requête.

```
SELECT DISTINCT nomVille, idVille  
FROM villes;
```

Une fois la ville choisie, le livreur clique sur ajouter, ce qui lance la requête suivante :

```
INSERT INTO livre VALUES (9, 3) ;
```

Ici l'idVille 9 correspond à Montargis

### Statut :

☒ En service

☐ Absent

Pour récupérer le statut actuel du livreur :

```
SELECT enService FROM livreurs  
WHERE idLivreur = 3;
```

Si le livreur change de statut en cochant une case :

```
UPDATE livreurs SET enService=true  
WHERE idLivreur = 3 ;
```

## Recherche restaurants :

Recherche restaurants

Recherche

sushi tacos

Tri

notes décroissant

Restaurants trouvés :

- Les délices d'asie
- O'Tacos
- Sushi à gogo
- Mexico Forever
- Le Jardin Gourmand

Page suivante >

La requête suivante permet de récupérer une liste de restaurant trié par **notes décroissantes** :

```
SELECT restaurants.nom,  
        AVG(commandes.note) AS avis  
FROM restaurants  
        NATURAL JOIN commandes  
        NATURAL JOIN associe  
        NATURAL JOIN mots_clefs  
WHERE restaurants.nom LIKE '%mot-clef%'  
        OR mots_clefs.mot LIKE '%mot-clef%'  
GROUP BY commande.idRestaurant, restaurants.nom  
ORDER BY AVG(commandes.note) DESC  
LIMIT 5;
```

Ce bouton permet de récupérer les 5 restaurants suivant en gardant les mêmes filtres via cette commande :

```
SELECT restaurants.nom,  
        AVG(commandes.note) AS avis  
FROM restaurants  
        NATURAL JOIN commandes  
        NATURAL JOIN associe  
        NATURAL JOIN mots_clefs  
WHERE restaurants.nom LIKE '%mot-clef%'  
        OR mots_clefs.mot LIKE '%mot-clef%'  
GROUP BY commande.idRestaurant, restaurants.nom  
ORDER BY AVG(commandes.note) DESC  
LIMIT 10  
OFFSET 5;
```