



2ème année Informatique

Projet SGBD

Flotte de vélos

21 mai 2021

Auteurs :

Quatadah Nasdami
Ghofrane Hamdouni
Doha Sadiq

Encadrant :

Sylvain Lombardy
Mohamed Mosbah

Table des matières

1	Introduction	2
2	Modélisation des données	2
2.1	Description du contexte de l'application	2
2.2	Modèle entité-association	3
2.3	Liste des opérations prévues sur la base	3
3	Schéma relationnel	4
3.1	Passage au relationnel	4
3.2	Schéma relationnel en 3e forme normale	4
3.3	Contraintes d'intégrité	5
3.4	Dépendances fonctionnelles	5
4	Implantation	6
4.1	Choix de la base SQL	6
4.2	Création de la base	6
4.3	Insertion de données	6
4.4	Mises à jour	6
4.5	Suppressions	6
4.6	Implémentation en SQL	7
5	Utilisation	8
5.1	Description de l'environnement d'exécution	8
5.2	Notice d'utilisation	8
5.3	Description de l'interface	10
5.3.1	Mode utilisateur	10
5.3.2	Mode administrateur	12
6	Conclusion	13

1 Introduction

L'objectif du projet est de mettre en oeuvre les notions et les méthodes vues dans le module SGBD. Le sujet choisi est ***Flotte de vélos***. Notre travail a été de modéliser, puis créer une base permettant la gestion des données pour suivre une flotte de vélos dans une métropole, en implémentant un certain nombre d'opérations à savoir la consultation et les mises à jour.

2 Modélisation des données

2.1 Description du contexte de l'application

On dispose des informations qui ont permis de construire un schéma conceptuel modélisé sous forme d'entités et associations donné par la figure 1.

Parmi les règles de gestion on peut citer:

- Un emprunt concerne un et un seul Vélo.
- Un emprunt concerne un et un seul Usager.
- Un usager peut faire plusieurs emprunts s'ils sont pas dans la même période.
- Un vélo se trouve dans une seule station.
- Un emprunt a exactement deux stations.
- La station d'arrivée d'un emprunt peut être nulle.
- La station d'arrivée d'un emprunt peut être la même que sa station de départ.
- Un vélo en cours d'utilisation a une station nulle.
- La distance est calculée entre exactement deux stations.
- Le nombre de vélos stockés dans une station ne peut pas dépasser son nombre de bornes.



On a mis en oeuvre quelques règles de gestion à l'aide des contraintes d'intégrités et les cardinalités.

Par exemple:

1. deux usagers ne peuvent pas avoir le même Nom, prénom, adresse, date d'adhésion

Ceci peut-être résolu par l'ajout d'un numéro d'utilisateur comme une clé unique et non nulle (contrainte d'intégrité).

2. Un emprunt concerne un et un seul Vélo, la cardinalité minimale et maximale seront égale à 1.

2.2 Modèle entité-association

Schéma conceptuel

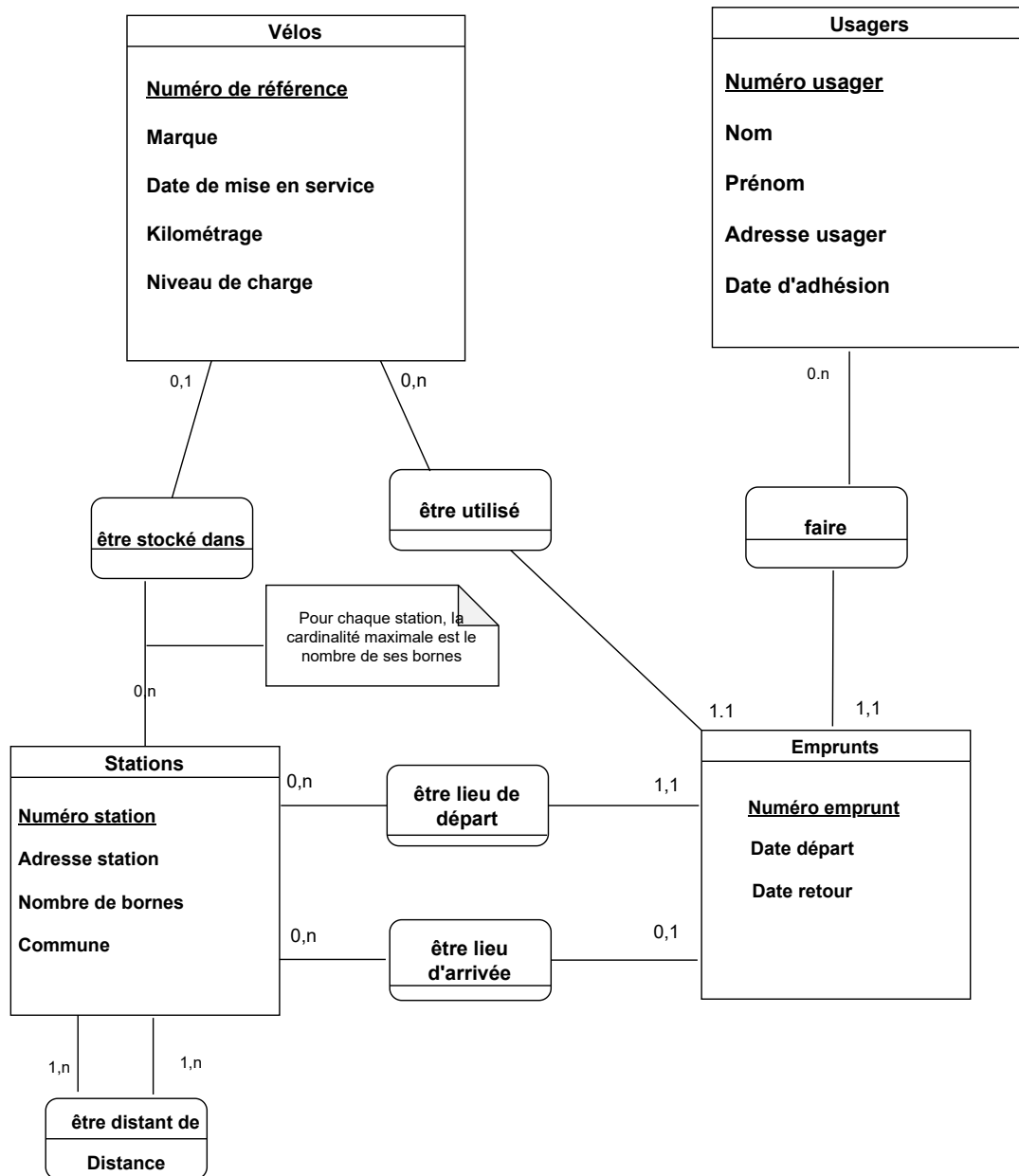


Figure 1: Modèle entité-association

2.3 Liste des opérations prévues sur la base

On a classé les opérations prévues sur la base suivant le mode utilisé: mode administrateur, mode utilisateur.

1. Mode administrateur

- Consultation des tables

- Ajout des colonnes dans les tables
- Suppression des colonnes des tables
- Modification des valeurs d'une table suite à la modification d'une autre
- Contraintes et vérifications de données
- Historisation des actions
- Modification des informations qui dépendent d'autres données

2. Mode utilisateur

- Ajout des colonnes dans les tables (notamment dans la table **Emprunts**)
- Mise à jour des colonnes ajoutées par l'utilisateur
- Modification d'informations, auxquelles il a accès, qui dépendent d'autres données (auxquelles il n'a pas accès)



Historisation des actions: On veut parfois garder une trace des actions effectuées sur la base c'est à dire qui a modifié telle ligne et quand



Pour mettre en oeuvre la modification des valeurs des colonne directement, la modification des valeurs d'une table suite à la modification d'une autre, les contraintes et les vérifications de données, l'historisation des actions, et la modification des informations qui dépendent d'autres données on a utilisé principalement les triggers

3 Schéma relationnel

3.1 Passage au relationnel

Pour passer du modèle conceptuel au modèle relationnel, on a procédé de la manières suivante:

- Les entités Vélos, Usagers, Stations et Emprunts deviennent des relations.
- Les associations *faire*, *être utilisé*, *être lieu de départ* et *être lieu d'arrivée* sont de type 1:n. Ainsi, les clés primaires des classes Usagers, Stations et Usagers deviennent des clés étrangères de la relation Emprunt.
- De même, pour l'association *être stocké dans*, la clé primaire de l'entité Stations devient une clé étrangère de la relation Velo.
- L'association *être distant de* est de type n:n. Il est nécessaire de définir donc une nouvelle relation qui aura comme clé primaire les clés étrangères des stations de départ et d'arrivée.

3.2 Schéma relationnel en 3e forme normale

Ainsi, on obtient le schéma relationnel suivant:

Velo(Numero reference, Marque, Date de mise en service, Kilometrage, Niveau de charge de la batterie, #Numero station)

Usager(Numero usager, Nom usager, Prenom usager, Adresse usager, Date adhésion)

Station(Numero station, Adresse station, Nombre bornes, Commune)

Emprunt(Numero emprunt, Date depart ,Date retour, #Numero usager (not null) , #Numero reference (not null), #Numero station depart (not null), #Numero station arrivee)

EtreDistant (#Numero station depart, #Numero station arrivee, Distance)

3.3 Contraintes d'intégrité

- **Primary key et foreign key:** Dans le schéma relationnel ci-dessus, les attributs soulignés représentent les clés primaires et les attributs précédés par # représentent les clés étrangères.
- **Null ou not null:** La clé étrangère Numéro station de la relation Velo peut être nulle dans le cas où le vélo sera emprunté et n'appartient donc à aucune station. Pour la relation Emprunt, le numéro d'utilisateur, de référence du vélo et de station de départ ne doivent pas être nulles puisque ces informations doivent nécessairement être renseignées dès l'ajout de l'emprunt. Toutefois, la station d'arrivée peut être nulle dans le cas où le vélo n'a pas encore été retourné par l'utilisateur.
- **Unique:** Pour des raisons de cohérence, l'attribut adresse de la table Station doit être unique. Toutefois, nous avons choisi de ne pas considérer cet attribut comme identifiant de la table car sa valeur peut être modifiée et parce qu'il ne peut pas être incrémenté automatiquement.
- **Check:** Il faut ajouter en plus des contraintes sur la validité des valeurs des attributs, comme le fait que le Kilométrage et le nombre de bornes doivent être positifs et que le niveau de charge de la batterie doit être entre 0 et 100.
- **Default:** Et finalement, certains attributs peuvent avoir des valeurs par défaut. Par exemple, pour les dates de mise en service des vélos, d'adhésion des usagers et de départ pour les emprunts, si au moment de l'insertion les valeurs de ces attributs n'ont pas été renseignées, la valeur par défaut sera la date du système au moment de l'exécution.

3.4 Dépendances fonctionnelles

Les différentes dépendances fonctionnelles sont résumées dans le graphe de couverture minimale suivant:

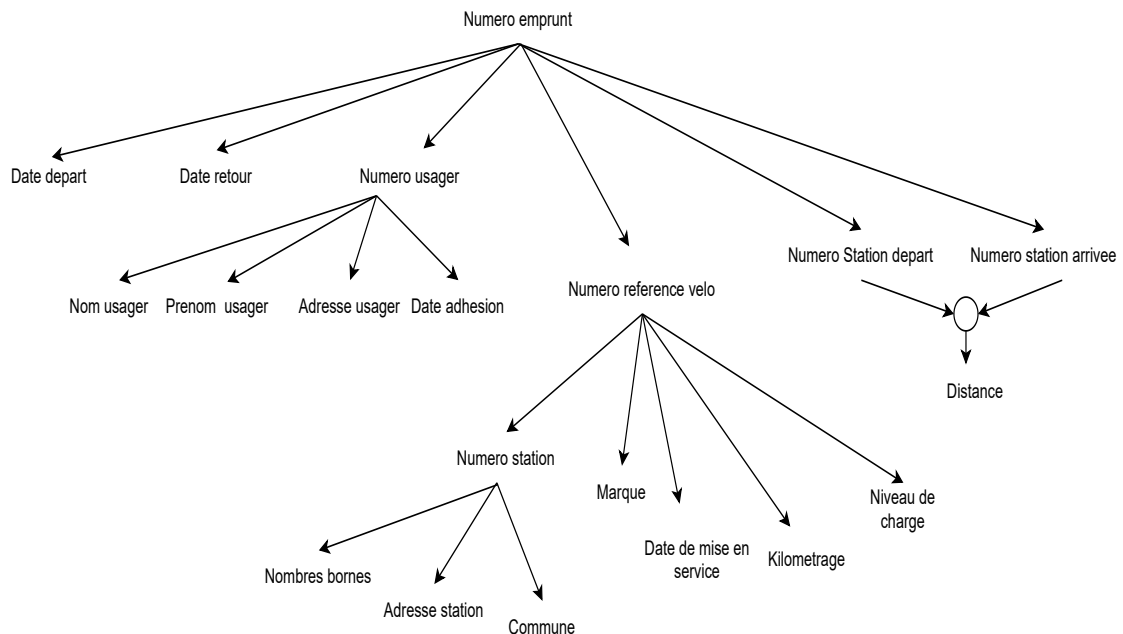


Figure 2: Graphe de couverture minimale

4 Implantation

4.1 Choix de la base SQL

Nous avons choisi de travailler avec MySQL principalement car il est gratuit, ce qui nous a permis de travailler de manière indépendante et plus efficace directement sur nos machines.

4.2 Création de la base

Nous avons créé la base **FLOTTE_DE_VELOS** contenant les tables déduites à partir des relations du schéma relationnel.

Pour chaque clé primaire, on ajoute la contrainte primary key et la commande *AUTO_INCREMENT* afin que cette clé soit incrémentée automatiquement à chaque ajout d'enregistrement.

En plus, pour chaque clé étrangère, on ajoute la contrainte *foreign key* qui permet d'assurer la liaison entre les différentes tables.

4.3 Insertion de données

Nous avons aussi ajouté plusieurs données à notre base afin de pouvoir bien tester les résultats des différentes requêtes implémentées.

Toutefois, pour s'assurer de la cohérence des différentes insertions que l'on effectue sur notre base, nous avons défini des triggers qui seront déclenchés avant toute insertion dans la table concernée. Par exemple, avant toute insertion d'un nouveau vélo dans la table **VELO**, il faut d'abord s'assurer que le nombre maximal de bornes de la station où on veut le déposer n'est pas atteint. Et avant de faire un emprunt, il faut vérifier que le vélo choisi se trouve bien dans la station de départ et que l'utilisateur qui veut effectuer l'emprunt n'a pas d'autre emprunt en cours.

4.4 Mises à jour

Pour effectuer des mises à jours sur les données déjà insérées dans la base, on a défini plusieurs procédures stockées.

Certaines de ces procédures sont appelées de manière automatique grâce à des déclencheurs après chaque mise à jour de la table concernée. Il s'agit des mises à jour qui sont fondamentales pour la cohérence de notre base de données, comme le fait qu'après chaque emprunt effectué, il faut changer la station du vélo et augmenter son kilométrage par la distance entre la station de départ et la station d'arrivée dans le cas où cette dernière n'a pas une valeur nulle.

D'autres procédures sont appelées à la demande de l'utilisateur comme pour modifier la station de retour ou la date de retour d'un vélo dans un emprunt lorsque ces attributs ont une valeur initiale nulle au moment de l'ajout de l'emprunt.

4.5 Suppressions

Pour supprimer une ligne d'une table, on a défini une procédure par table qui prend en paramètre un identifiant et supprime la ligne correspondante. Toutefois, pour assurer le bon fonctionnement de la base, il est nécessaire avant de supprimer un objet de supprimer tous les objets qui lui sont reliés. Ainsi, pour supprimer par exemple un usager, il faut d'abord supprimer tous les emprunts qu'il a effectués. Pour assurer ceci, on a utilisé des triggers.

Supprimer la base entière revient à supprimer toutes les tables, les données, les index, les contraintes ainsi que les procédures stockées de la base.

4.6 Implémentation en SQL

Ainsi, pour implémenter les différentes fonctionnalités en SQL, on a eu recours à différents outils:

- Des requêtes notamment pour la création et la suppression de la base, la consultation des données et les statistiques.

```
SELECT VELO.*  
from VELO inner join EMPRUNT  
on VELO.NUMERO_REFERENCE=EMPRUNT.NUMERO_REFERENCE  
where EMPRUNT.DATE_RETOUR IS NULL;
```

Par exemple, cette requête permet de donner la liste des vélos en cours d'utilisation. Cette propriété est vérifiée par tous les vélos dont la date de retour correspond à NULL, ce qui signifie qu'ils n'ont pas encore été retournés.

- Des procédures pour réaliser les actions qui seront répétées plusieurs fois avec des paramètres de valeurs différentes tout au long de la manipulation de notre base de données principalement les mises à jour.

```
DELIMITER //  
  
CREATE PROCEDURE changer_station_velo(station INT, reference INT  
)  
BEGIN  
    update VELO  
    SET VELO.NUMERO_STATION=station  
    WHERE VELO.NUMERO_REFERENCE=reference;  
END //
```

La procédure ci-dessus permet de changer la station d'un vélo, et ce en exécutant la requête située entre BEGIN et END en utilisant les paramètres *station* et *reference* passés au moment de l'appel à cette procédure.

- Et finalement, les triggers pour les actions qui vont être déclenchés automatiquement à la suite d'un événement, et qu'il ne faut surtout pas oublier de réaliser pour préserver la cohérence de la base.

```
DELIMITER //  
  
create or replace trigger AFTER_UPDATE_EMPRUNT  
after UPDATE on EMPRUNT  
for each row  
BEGIN  
if new.NUMERO_STATION_ARRIVEE!=old.NUMERO_STATION_ARRIVEE then  
    CALL changer_station_velo(new.NUMERO_STATION_ARRIVEE,new.  
        NUMERO_REFERENCE);  
    CALL augmenter_kilometrage_velo(new.NUMERO_STATION_DEPART,new.  
        .NUMERO_STATION_ARRIVEE,new.NUMERO_REFERENCE);  
end if;  
end //
```


Ce trigger est déclenché après chaque mise à jour de la table EMPRUNT. Pour chaque ligne qui a été modifiée, si c'est la colonne *NUMERO_STATION_ARRIVEE* qui est concernée par cette modification, on appelle la procédure *changer_station_velo* expliquée auparavant, puis la procédure *augmenter_kilometrage_velo* qui va modifier le kilométrage du vélo concerné par l'emprunt.

5 Utilisation

5.1 Description de l'environnement d'exécution

Pour réaliser l'interface, nous avons utilisé l'environnement de développement **Xampp** qui permet de manipuler la base de données en utilisant le langage **PHP** qui aussi été choisi pour sa popularité dans ce domaine et sa simplicité.

5.2 Notice d'utilisation

- Aller vers le lien suivant <https://www.apachefriends.org/fr/index.html> et suivre les étapes d'installation.
- Sur le terminal, dans le répertoire où vous l'avez téléchargé, exécuter **sudo ./xampp.run**
- Puis exécuter la commande suivante: **sudo /opt/lampp/lampp start**
- Aller vers le lien <http://localhost/phpmyadmin/>



Figure 3: Importation de la base

1. Sélectionner **Importer**
2. Parcourir le fichier *creation.sql*
3. Cliquer sur exécuter
4. Après l'exécution, retourner à la base de données en cliquant sur **Base de données**. Vous devez obtenir le résultat suivant:

À partir de l'étape 7, vous pouvez directement utiliser l'interface réalisée (voir section *Description interface*). Pour accéder à cette interface, à partir du répertoire contenant les sources de notre projet, exécuter la commande suivante: **sudo cp -r Application/ /opt/lampp/htdocs/** Ensuite, aller vers le lien <http://localhost/Application/>.

5.3 Description de l'interface

5.3.1 Mode utilisateur

Nous avons mis en place un système de connexion qui permet à un utilisateur de se connecter directement à son compte en saisissant son identifiant. De plus, il est possible pour un nouvel utilisateur de créer un nouveau compte en remplissant un formulaire contenant son nom, son prénom et son adresse. Après la création du compte, les informations de l'utilisateur seront ajoutées à la base de données et son identifiant lui sera affiché pour qu'il puisse se connecter à son compte.

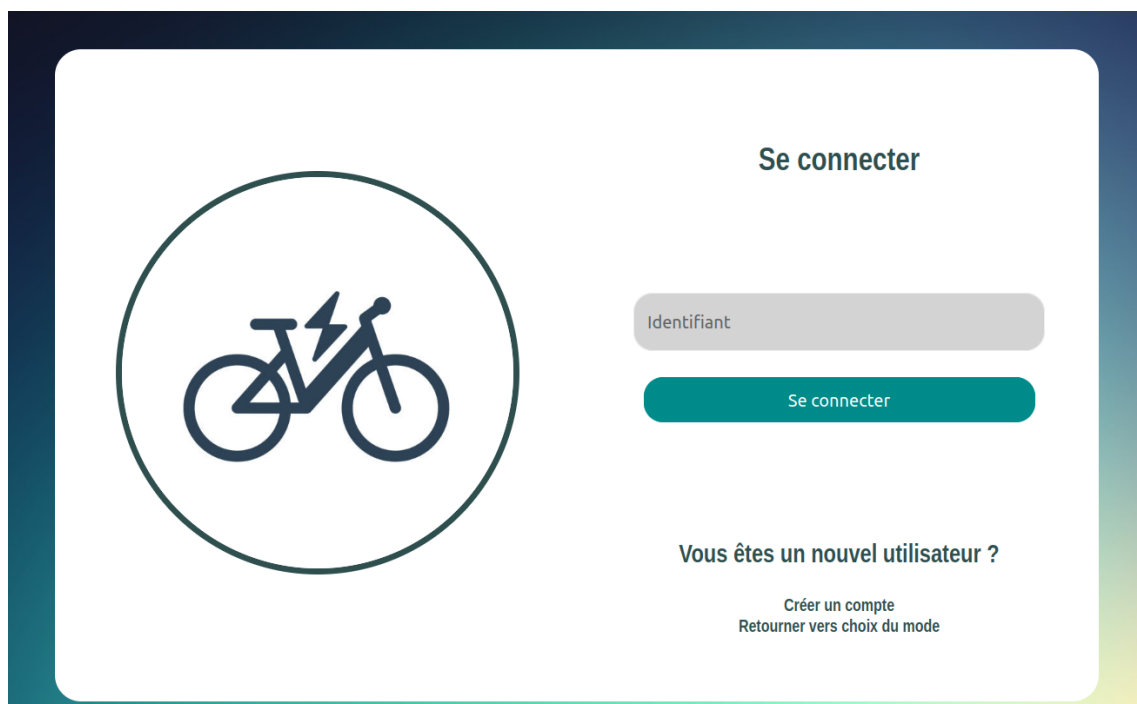


Figure 7: Page d'accueil

Une fois l'utilisateur se connecte, il aura à choisir une commune et puis l'ensemble des stations de cette commune lui sera affiché comme dans la figure suivant :

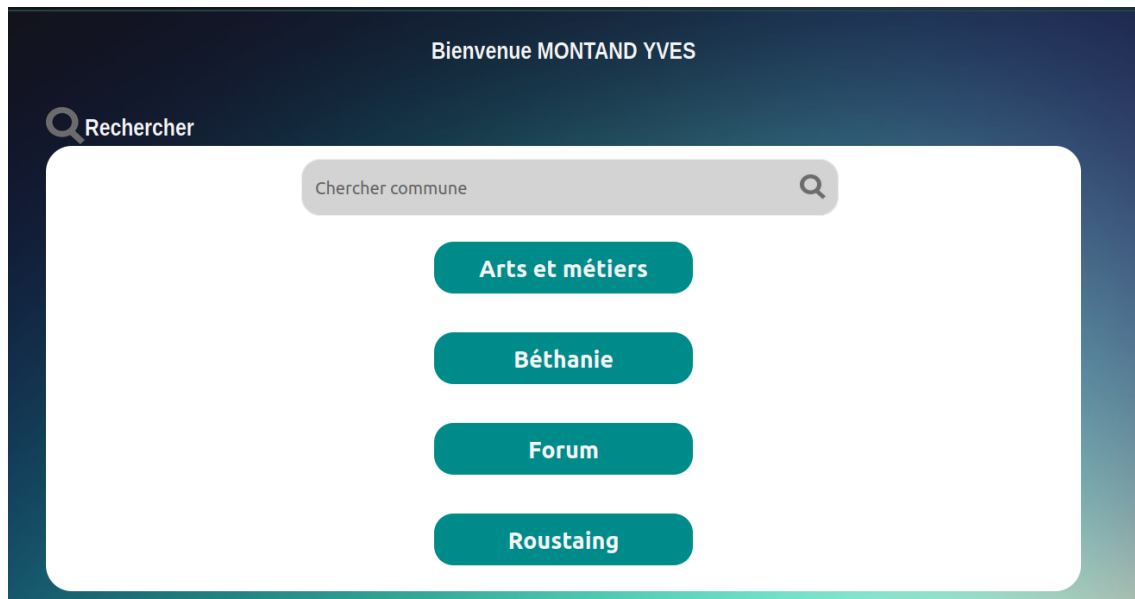


Figure 8: Recherche de station dans une commune entrée

Et lorsqu'il choisit la station, il obtiendra le nombre de vélos disponibles dans cette station et la liste de ces vélos. Il pourra par la suite choisir le vélo qu'il veut emprunter.

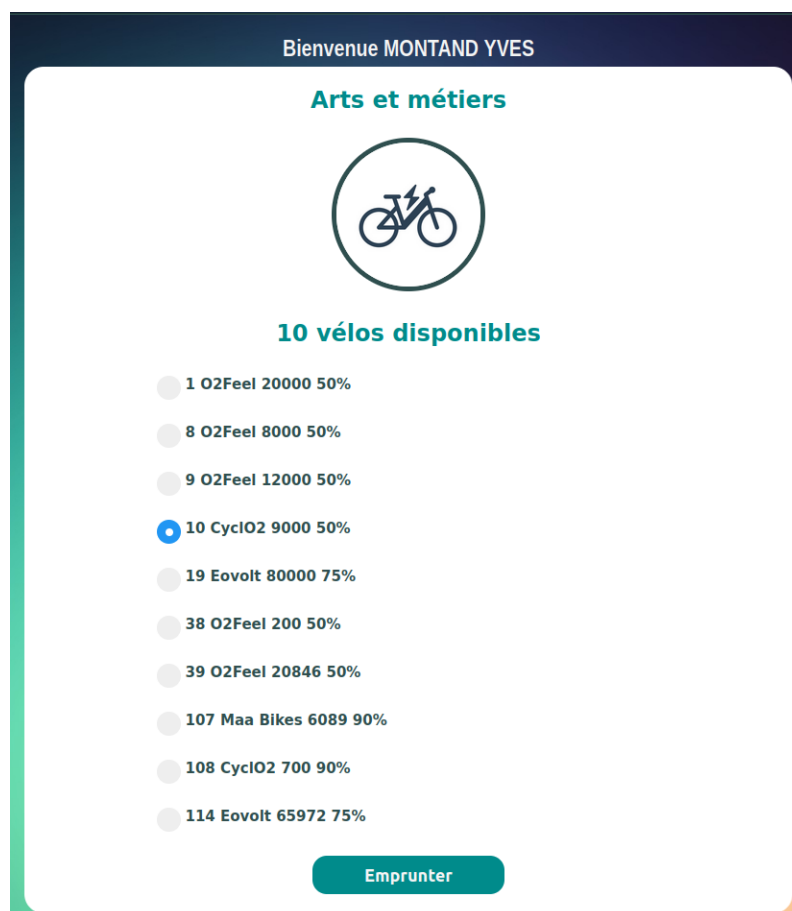



Figure 9: Vélos disponibles

Un nouveau emprunt sera ajouté à la base de données en prenant comme date de départ la date du système au moment de l'emprunt et en gardant NULL dans la date et station de retour.

Bienvenue MONTAND YVES

Votre emprunt actuel



Data emprunt : 2021-12-09 06:33:20

Numéro vélo : 10

Emprunté de la station : Arts et métiers

Arts et métiers

Déposer

- Arts et métiers
- Béthanie
- Forum
- Roustaing
- France Alouette
- Bergnonié**
- Saint-Nicolas
- Victoire
- Musée daquitaine
- Hôtel de ville
- Gambetta
- Grand théâtre

Figure 10: Récapitulatif d'emprunt

Lorsqu'il veut déposer le vélo, l'utilisateur sélectionne la station de retour, et l'emprunt sera mis à jour dans la base de données. Il sera redirigé vers la page de choix de commune s'il veut effectuer un nouvel emprunt, sinon il peut se déconnecter.

5.3.2 Mode administrateur

Il est également possible de se connecter en mode administrateur pour pouvoir accéder à l'ensemble des données de la base ou les mettre à jour et consulter les statistiques.

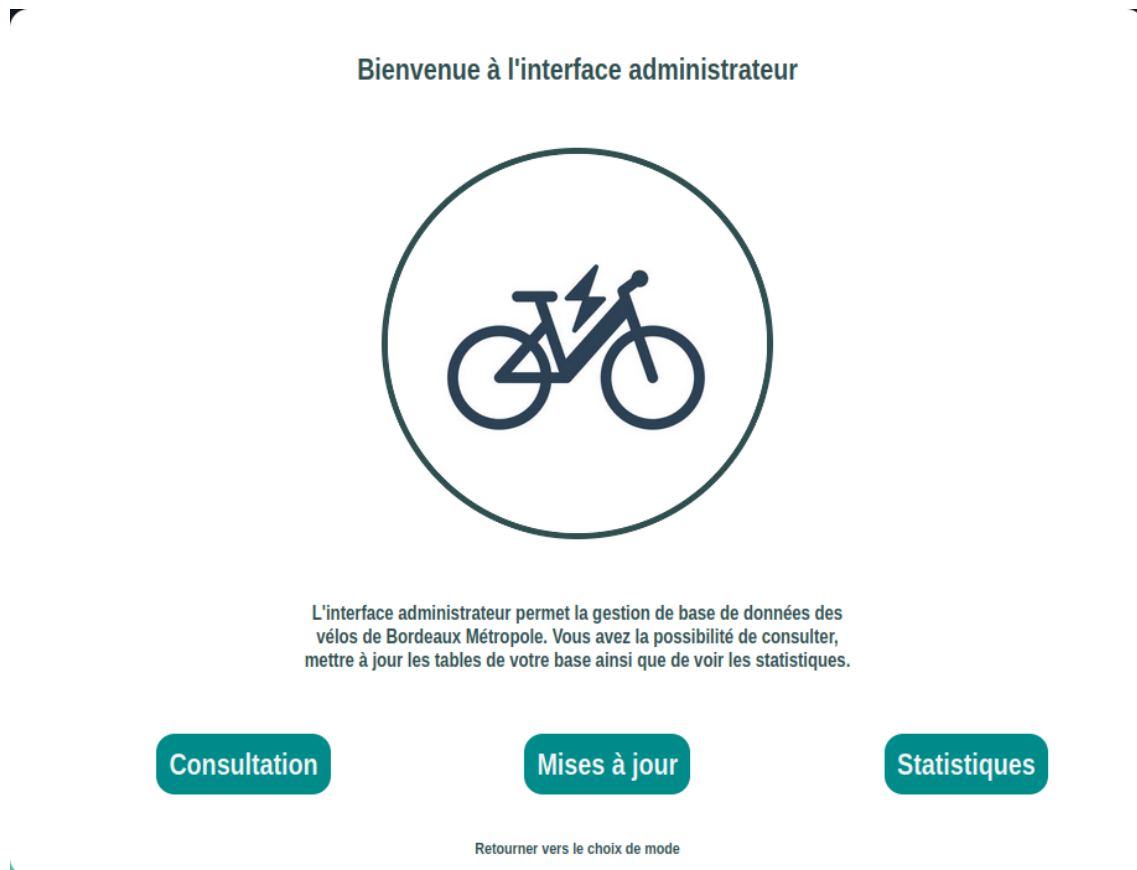


Figure 11: Interface administrateur

Pour la consultation des données, l'administrateur peut sélectionner le nom de la table qu'il veut consulter et le résultat sera affiché sous forme de tableau. Pour les mises à jour, il est possible d'ajouter un vélo ou une station ou de les supprimer. Et pour les statistiques, il a accès aux résultats suivants:

- Moyenne du nombre d'utilisateurs par vélo par jour.
- Moyenne des distances parcourues par les vélos sur une semaine.
- Classement des stations par nombre de places disponibles par commune.
- Classement des vélos les plus chargés par station.

6 Conclusion

Ce projet nous a permis de renforcer nos connaissances en systèmes de gestion de base de données et ceci en les mettant en oeuvre. En allant du schéma conceptuel à une interface proposée à l'utilisateur et à l'administrateur, et en passant par le schéma relationnel, nous avons pu faire d'un énorme pas vers le monde de l'informatique.