# La base de données

*Back-end*

## Les data récupérés

Pour notre base de données, nous avons récupéré 4 JSONs :
- les velibs
- les trilibs
- les bornes électriques
- les trimobiles

## Les velibs

Pour les velibs nous avons dans notre table :
- les states
- les freedock
- les credit_card
- les station_name
- la latitudde
- la longitudde
- les bike_available

## Les colonnes que l'on utilise déjà dans notre projet

Pour pouvoir calculer la distance entre les velibs et nos monuments nous avons gardé leurs coordonnées GPS, soit la latitude et la longitude. Ils seront utiles afin d'afficher la position de la station sur Google map.

Afin d'afficher le velib, nous avons gardé le "station_name" qui sera le nom de la station affiché sur notre projet.

## Les colonnes qui seront utiles lors de l'évolution du projet

Les "freedock" permettent d'afficher le nombre de docks disponibles pour garer son velib.
Les "credit_card" sont des booléens qui affichent si la station prend la carte de crédit.
Les "states" affichent l'état de la station, si elle marche, si elle est fermée ou bien si elle est en cours de construction.
Les "bike_available" permettent d'afficher le nombre de velibs disponibles dans la station.

## Les trilibs

Pour les trilibs nous avons dans notre table
- les latitudes
- les longitudes
- les wastetype
- les adresse
- les city
- les zipcode

## Les colonnes que l'on utilise déjà dans notre projet

Pour pouvoir calculer la distance entre les trilibs et nos monuments nous avons gardé leurs coordonnées GPS, soit la latitude et la longitude. Ils permettent d'afficher la position du trilib sur Google map.

Les “adresses”, les “city” et les “zipcode” permettent d’afficher l’adresse complète sur notre projet.

## Les colonnes qui seront utiles lors de l’évolution du projet

Les “wastetype” nous serrons utiles à l’avenir afin d’afficher le type de déchet que doit contenir la poubelle, si c’est du verre, du plastique des cannettes...

## Les bornes électriques

Pour les bornes, nous avons dans notre table :
- les schedule
- les aftersales_phone
- les latitudes
- les longitudes
- les electric_type
- les connector_type
- les city
- les adresses
- les zipcode
- les watts
- les station_name

## Les colonnes que l’on utilise déjà dans notre projet

Pour pouvoir calculer la distance entre les bornes et nos monuments nous avons gardé les coordonnées GPS, soit la latitude et la longitude. Ils permettent d’afficher la position de la borne sur Google map.

Les “adresses”, les “city” et les “zipcode” nous sont utiles pour afficher l’adresse complète sur notre projet. La “station_name” affiche le nom de la station.

## Les colonnes qui seront utiles lors de l’évolution du projet

Les “schedule” nous serrons utiles pour afficher les jours et les heures de disponibilité des bornes.
Les “aftersales_phone” nous serrons utiles pour afficher le numéro du SAV des bornes électriques.
Les “electric_type”, les “connector_type” et les “watt” nous serrons utiles pour afficher le type de prise et le type de courant que fournissent les bornes.

## Les trimobiles

Pour les trimobiles nous avons dans notre table :
- les city
- les zipcode
- les adresses
- les schedule
- les longitudes
- les latitudes
- les time_range
- les adress_supplement

## Les colonnes que l’on utilise déjà dans notre projet

Pour pouvoir calculer la distance entre les trimobiles et nos monuments nous avons gardé les coordonnés GPS, soit la latitude et la longitude. Ils permettent d’afficher la position du trimobile sur Google map.

Les “adresses”, les “city”, les “zipcode” et les “address_supplement” nous sont utiles pour afficher l’adresse complète sur notre projet.

### Les colonnes qui seront utiles lors de l’évolution du projet

Les “schedule” et les “time_range” nous servirons à afficher les jours et les heures où le trimobile est présent.

## Les monuments

Les monuments ont des datas créées par nous-mêmes après quelques recherches sur Internet et Google map, elle contiennent:
- le nom du monument
- sa longitude et sa latitude
- son adresse
- sa ville
- son code postal
- son URL pour son image
- le sport qui va se dérouler à cet endroit

## Les tables “_distance_monument”

Elles sont générées entre la table monuments et respectivement “trilib, velib, trimobile, electricterminal”, les données "distance_km" sont calculées par notre script PHP qui les injecte
grâce aux longitudes et aux latitudes des autres tables.

## Relation ManyToMany en ManyToOne

Comme vous pouvez le constater, il y a une différence entre le MCD de JMerise et le MCD que nous avons fait. Le MCD que nous avons fait contient des ManyToMany entre les monuments et les autres tables alors que celui de JMerise lui contient des tables en plus (les fameuses tables “_distance_monument”) avec des relations ManyToOne.

Cela est dû à Symfony et Doctrine, nous avons choisi de créer notre base de données avant le Symfony et de génèrer nos entités par rapport à la base (mapping) et non par Symfony. Doctrine n’aime pas les associations contenant des datas (distance_km) nous avons du décomposer nos associations en table associative.

## La création de la base de données

La base de données a été créée avec le logiciel JMerise et les données ont été insérées grâce à un script PHP qui met en forme nos données contenues dans des JSONs. Pour simplifier le travail de nos FRONT, un dump de la base de données finalisé a été créé pour l’installer avec une seule commande `mysql -uroot -p < DB.sql`.

## Choix techniques API

Nous avons décidé de créer la base de données sans utiliser Doctrine, pour avoir un contrôle total de la base de données.

Pour cela après avoir généré la bdd, nous l'avons importé sur notre projet Symfony à l'aide de la commande :

(Plus en détail sur le README)

./bin/console doctrine:mapping:import "App\Entity" annotation --path=src/Entity

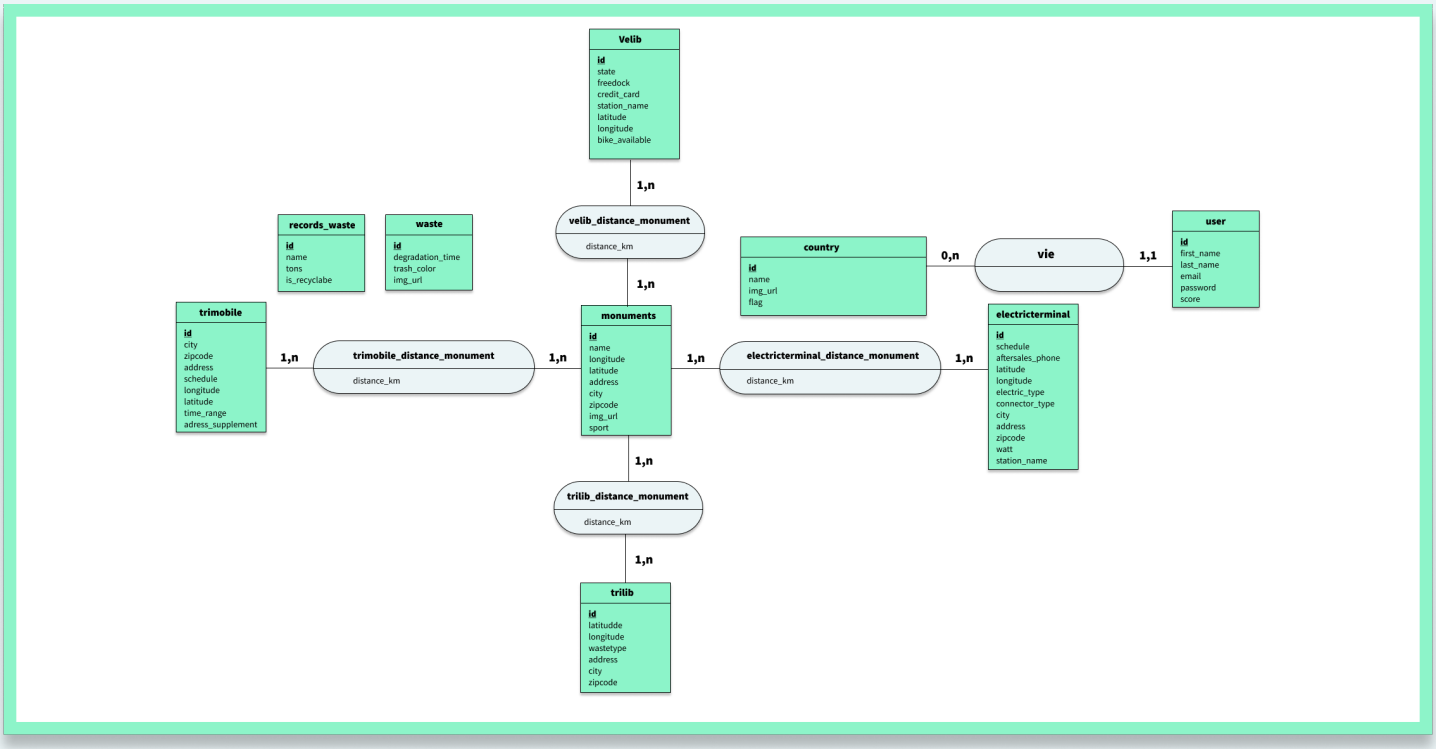
Nous avons également mis en place un système de variable d'environnement pour sécurisé l'accès à la bdd ainsi qu'au BOT Twitter. Rajouté à ça nous avons Dockerisé notre application pour minimiser les problèmes de dépendances.

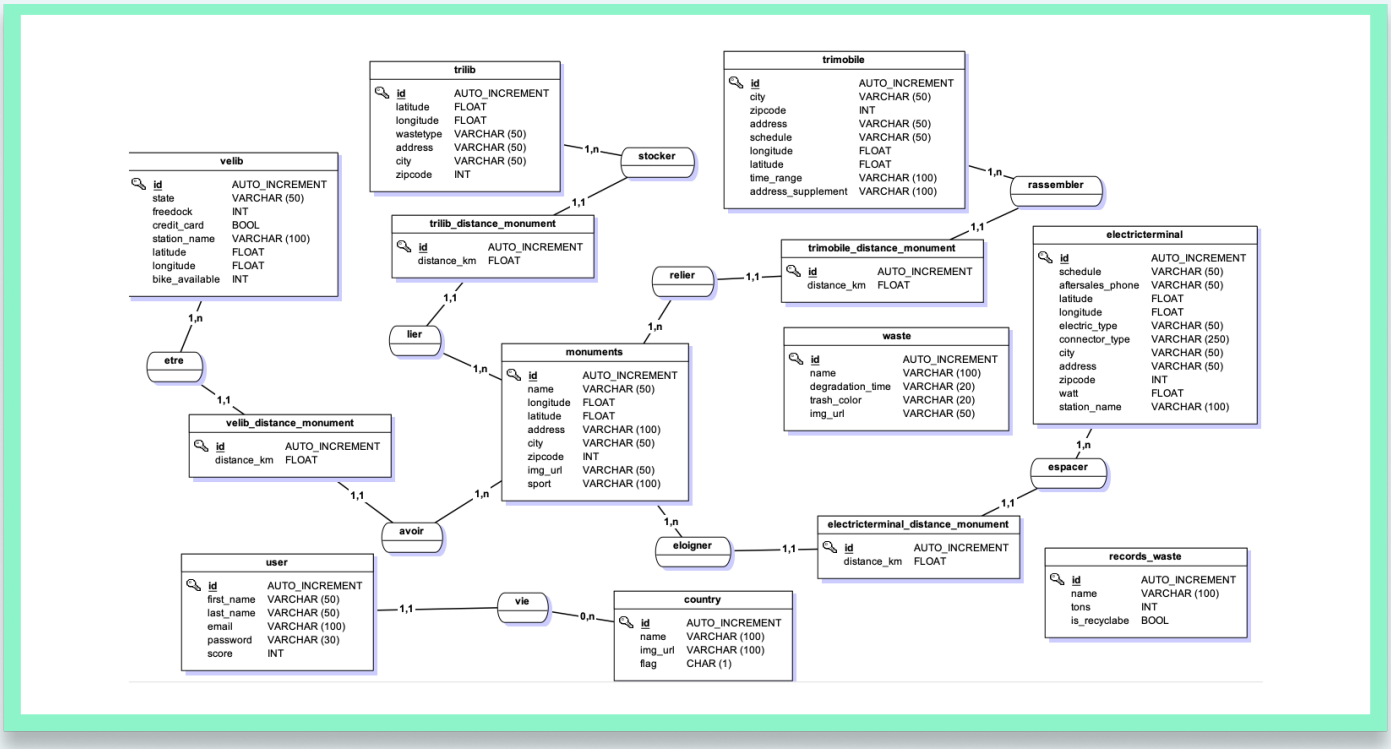
## Liste de packages Symfony

- abraham/twitteroauth : Librairie PHP pour l'API Twitter
- nelmio/cors-bundle : Système de management de CORS
- symfony/dotenv : Système de management des variables d'environnement
- symfony/orm-pack : Doctrine ORM
- symfony/yaml : YAML

# Annexes

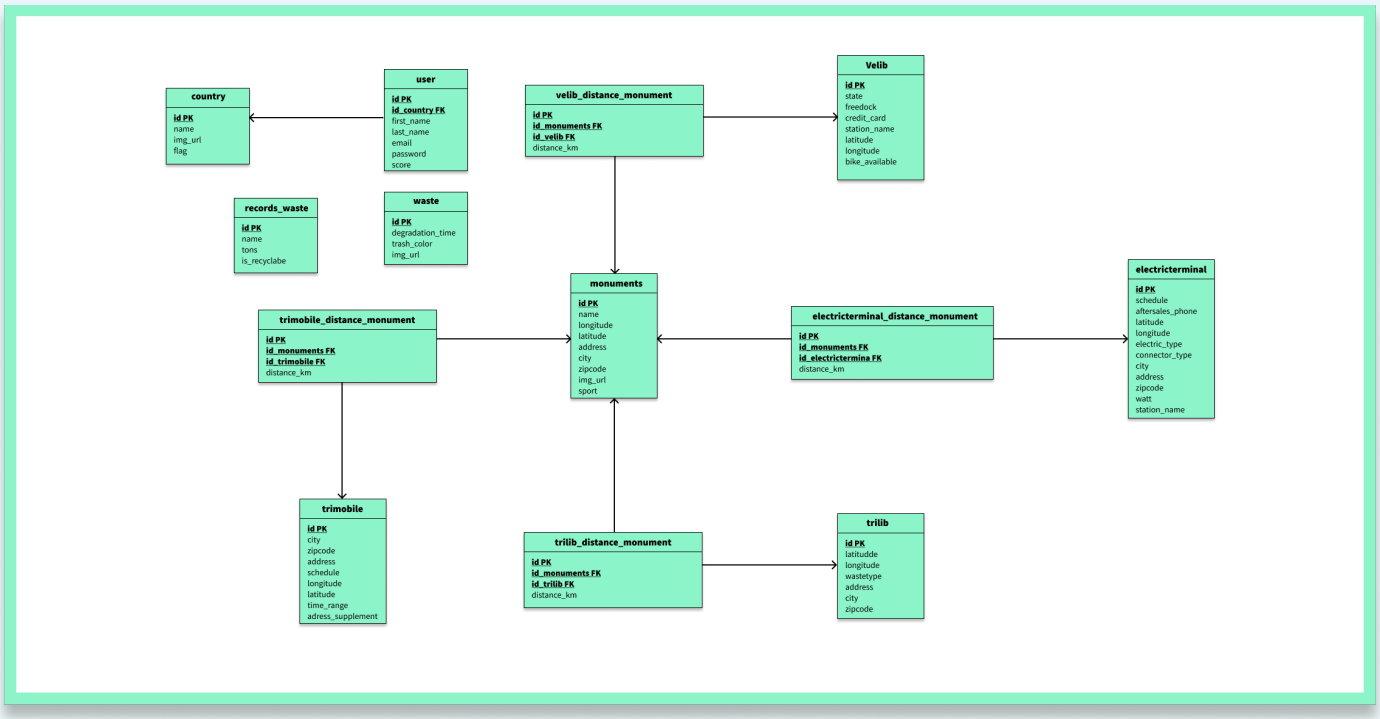
## MCD

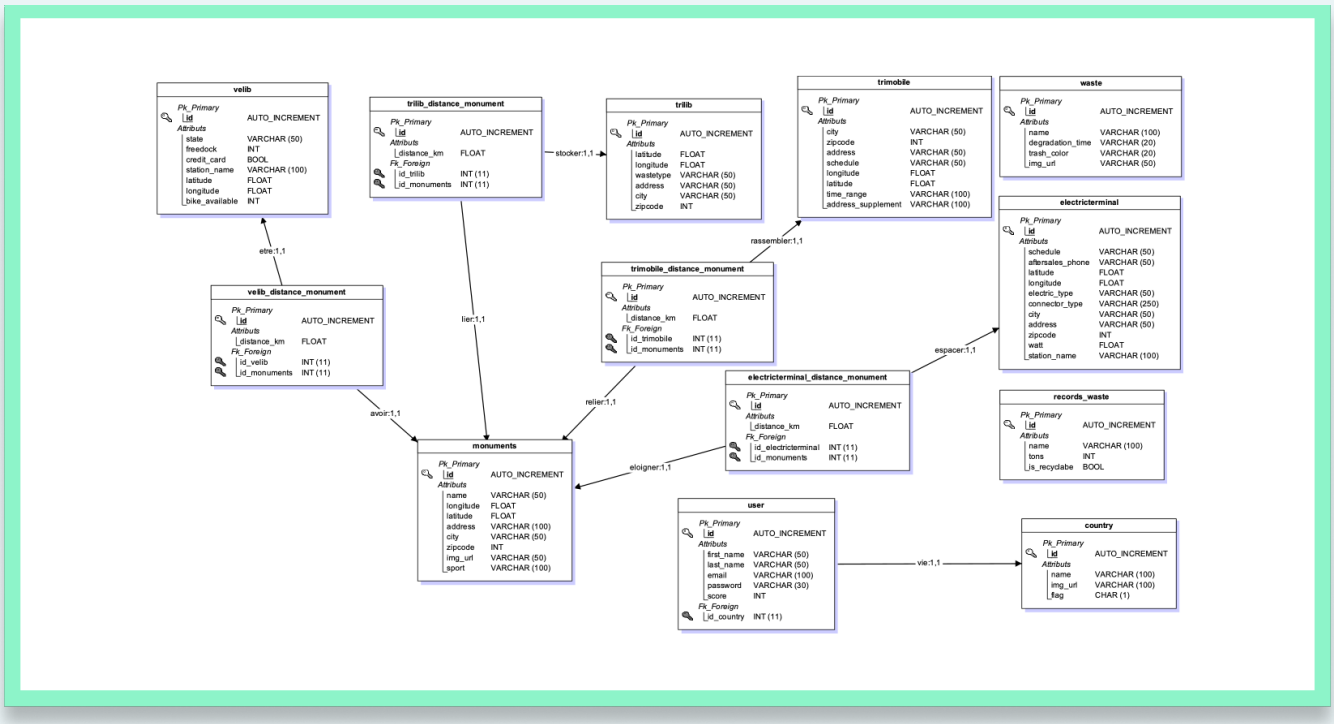

Velib
id
state
freedock
credit_card
station_name
latitude
longitude
bike_available
1,n
velib_distance_monument
distance_km
1,n
records_waste
id
name
tons
is_recyclabe
waste
id
degradation_time
trash_color
img_url
country
id
name
img_url
flag
0,n
vie
1,1
user
id
first_name
last_name
email
password
score
trimobile
id
city
zipcode
address
schedule
longitude
latitude
time_range
adress_supplement
1,n
trimobile_distance_monument
distance_km
1,n
monuments
id
name
longitude
latitude
address
city
zipcode
img_url
sport
1,n
electricterminal_distance_monument
distance_km
1,n
electricterminal
id
schedule
aftersales_phone
latitude
longitude
electric_type
connector_type
city
address
zipcode
watt
station_name
1,n
trilib_distance_monument
distance_km
1,n
trilib
id
latitudde
longitude
wastetype
address
city
zipcode


trilib
id AUTO_INCREMENT
latitude FLOAT
longitude FLOAT
wastetype VARCHAR (50)
address VARCHAR (50)
city VARCHAR (50)
zipcode INT
1,n
stocker
1,1
trimobile
id AUTO_INCREMENT
city VARCHAR (50)
zipcode INT
address VARCHAR (50)
schedule VARCHAR (50)
longitude FLOAT
latitude FLOAT
time_range VARCHAR (100)
address_supplement VARCHAR (100)
1,n
rassembler
velib
id AUTO_INCREMENT
state VARCHAR (50)
freedock INT
credit_card BOOL
station_name VARCHAR (100)
latitude FLOAT
longitude FLOAT
bike_available INT
trilib_distance_monument
id AUTO_INCREMENT
distance_km FLOAT
1,1
trimobile_distance_monument
id AUTO_INCREMENT
distance_km FLOAT
1,1
relier
1,1
electricterminal
id AUTO_INCREMENT
schedule VARCHAR (50)
aftersales_phone VARCHAR (50)
latitude FLOAT
longitude FLOAT
electric_type VARCHAR (50)
connector_type VARCHAR (250)
city VARCHAR (50)
address VARCHAR (50)
zipcode INT
watt FLOAT
station_name VARCHAR (100)
1,n
lier
1,1
1,n
etre
1,n
monuments
id AUTO_INCREMENT
name VARCHAR (50)
longitude FLOAT
latitude FLOAT
address VARCHAR (100)
city VARCHAR (50)
zipcode INT
img_url VARCHAR (50)
sport VARCHAR (100)
1,n
waste
id AUTO_INCREMENT
name VARCHAR (100)
degradation_time VARCHAR (20)
trash_color VARCHAR (20)
img_url VARCHAR (50)
1,1
velib_distance_monument
id AUTO_INCREMENT
distance_km FLOAT
1,n
espacer
1,1
1,1
avoir
1,n
1,n
eloigner
1,1
electricterminal_distance_monument
id AUTO_INCREMENT
distance_km FLOAT
user
id AUTO_INCREMENT
first_name VARCHAR (50)
last_name VARCHAR (50)
email VARCHAR (100)
password VARCHAR (30)
score INT
1,1
vie
0,n
country
id AUTO_INCREMENT
name VARCHAR (100)
img_url VARCHAR (100)
flag CHAR (1)
records_waste
id AUTO_INCREMENT
name VARCHAR (100)
tons INT
is_recyclabe BOOL

# Annexes

## MLD


country
id PK
name
img_url
flag
user
id PK
id_country FK
first_name
last_name
email
password
score
velib_distance_monument
id PK
id_monuments FK
id_velib FK
distance_km
Velib
id PK
state
freedock
credit_card
station_name
latitude
longitude
bike_available
records_waste
id PK
name
tons
is_recyclabe
waste
id PK
degradation_time
trash_color
img_url
monuments
id PK
name
longitude
latitude
address
city
zipcode
img_url
sport
electricterminal
id PK
schedule
aftersales_phone
latitude
longitude
electric_type
connector_type
city
address
zipcode
watt
station_name
trimobile_distance_monument
id PK
id_monuments FK
id_trimobile FK
distance_km
electricterminal_distance_monument
id PK
id_monuments FK
id_electricermina FK
distance_km
trimobile
id PK
city
zipcode
address
schedule
longitude
latitude
time_range
adress_supplement
trilib_distance_monument
id PK
id_monuments FK
id_trilib FK
distance_km
trilib
id PK
latitudde
longitude
wastetype
address
city
zipcode


velib
Pk_Primary
id AUTO_INCREMENT
Attributs
state VARCHAR (50)
freedock INT
credit_card BOOL
station_name VARCHAR (100)
latitude FLOAT
longitude FLOAT
bike_available INT
trilib_distance_monument
Pk_Primary
id AUTO_INCREMENT
Attributs
distance_km FLOAT
Fk_Foreign
id_trilib INT (11)
id_monuments INT (11)
stocker:1,1
trilib
Pk_Primary
id AUTO_INCREMENT
Attributs
latitude FLOAT
longitude FLOAT
wastetype VARCHAR (50)
address VARCHAR (50)
city VARCHAR (50)
zipcode INT
trimobile
Pk_Primary
id AUTO_INCREMENT
Attributs
city VARCHAR (50)
zipcode INT
address VARCHAR (50)
schedule VARCHAR (50)
longitude FLOAT
latitude FLOAT
time_range VARCHAR (100)
address_supplement VARCHAR (100)
waste
Pk_Primary
id AUTO_INCREMENT
Attributs
name VARCHAR (100)
degradation_time VARCHAR (20)
trash_color VARCHAR (20)
img_url VARCHAR (50)
electricterminal
Pk_Primary
id AUTO_INCREMENT
Attributs
schedule VARCHAR (50)
aftersales_phone VARCHAR (50)
latitude FLOAT
longitude FLOAT
electric_type VARCHAR (50)
connector_type VARCHAR (250)
city VARCHAR (50)
address VARCHAR (50)
zipcode INT
watt FLOAT
station_name VARCHAR (100)
ebte:1,1
velib_distance_monument
Pk_Primary
id AUTO_INCREMENT
Attributs
distance_km FLOAT
Fk_Foreign
id_velib INT (11)
id_monuments INT (11)
lier:1,1
rassembler:1,1
trimobile_distance_monument
Pk_Primary
id AUTO_INCREMENT
Attributs
distance_km FLOAT
Fk_Foreign
id_trimobile INT (11)
id_monuments INT (11)
espacer:1,1
avoir:1,1
relier:1,1
electricterminal_distance_monument
Pk_Primary
id AUTO_INCREMENT
Attributs
distance_km FLOAT
Fk_Foreign
id_electricterminal INT (11)
id_monuments INT (11)
records_waste
Pk_Primary
id AUTO_INCREMENT
Attributs
name VARCHAR (100)
tons INT
is_recyclabe BOOL
eloigner:1,1
monuments
Pk_Primary
id AUTO_INCREMENT
Attributs
name VARCHAR (50)
longitude FLOAT
latitude FLOAT
address VARCHAR (100)
city VARCHAR (50)
zipcode INT
img_url VARCHAR (50)
sport VARCHAR (100)
user
Pk_Primary
id AUTO_INCREMENT
Attributs
first_name VARCHAR (50)
last_name VARCHAR (50)
email VARCHAR (100)
password VARCHAR (30)
score INT
Fk_Foreign
id_country INT (11)
vie:1,1
country
Pk_Primary
id AUTO_INCREMENT
Attributs
name VARCHAR (100)
img_url VARCHAR (100)
flag CHAR (1)