Projet BD Réseau - Version finale

Numéro du groupe : Groupe A1

Clément GOUBERT

Lounis BOUHADOUN

Yige YANG

Salle de sport

CY Cergy Paris Université - Département Sciences Informatiques
2023/2024, UE Projet BD / Réseau

27/11/2023

Tables des matières

1.Présentation du Contexte (Contexte de présentation) :

Notre projet consiste en la mise en place d'un système de gestion de salle de sport que nous avons décidé de nommer **muscleUP**. Le système doit fournir un ensemble de fonctions pour permettre le bon le bon déroulement des différentes étapes du fonctionnement de la salle de sport, comme l'inscription des personnes, l'accès des différentes personnes qui fréquentent la salle, l'accès aux cours ainsi qu'au parking. Nous devons donc gérer ces différents informations pour assurer le bon fonctionnement de notre projet

Notre projet devra donc gérer :

- L'inscription d'une nouvelle personne personne, et de son ajout à la base de données avec ces informations personnelles et l'attribution d'un forfait
- La gestions des différents accès membre/employés
- La gestion du parking

Ce projet a donc pour objectif la conception d'une base de données et la création d'une communication client/serveur au sain de la salle de sport avec les échanges réseau associés.

2.Le planning

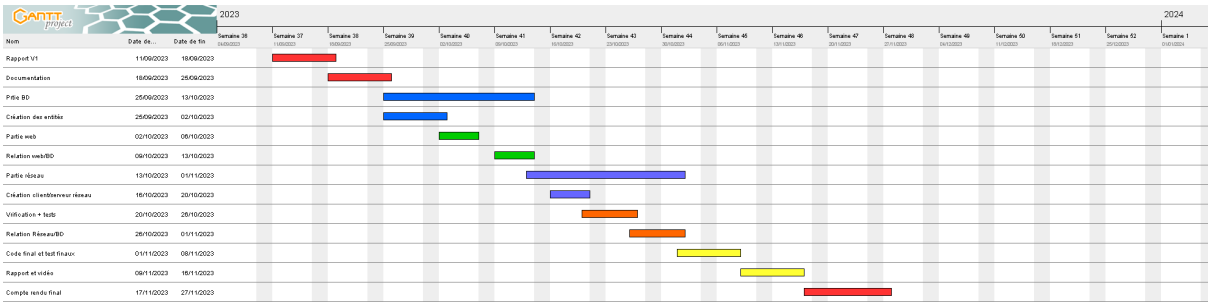


Diagramme de GANTT

3. Specification du projet

Dans cette partie nous allons aborder les notions importantes de notre projet:

3.1 Architecture du projet

Échanges réseau entre le client et le serveur: pour le projet nous avons décidé de coder le client en java et le serveur en python:

- **Client réseau:** est un logiciel qui envoie des demandes à un serveur. Dans notre cas il est représenté par un scan sur lequel le membre inscrit à la salle, où ce dernier va scanner sa carte membre pour pouvoir accéder à l'intérieur de la salle
- **Serveur réseau:** est un dispositif informatique qui offre des services à un ou plusieurs clients. Dans notre projet, le serveur réseau gère une connexion à la fois des clients à ce serveur.

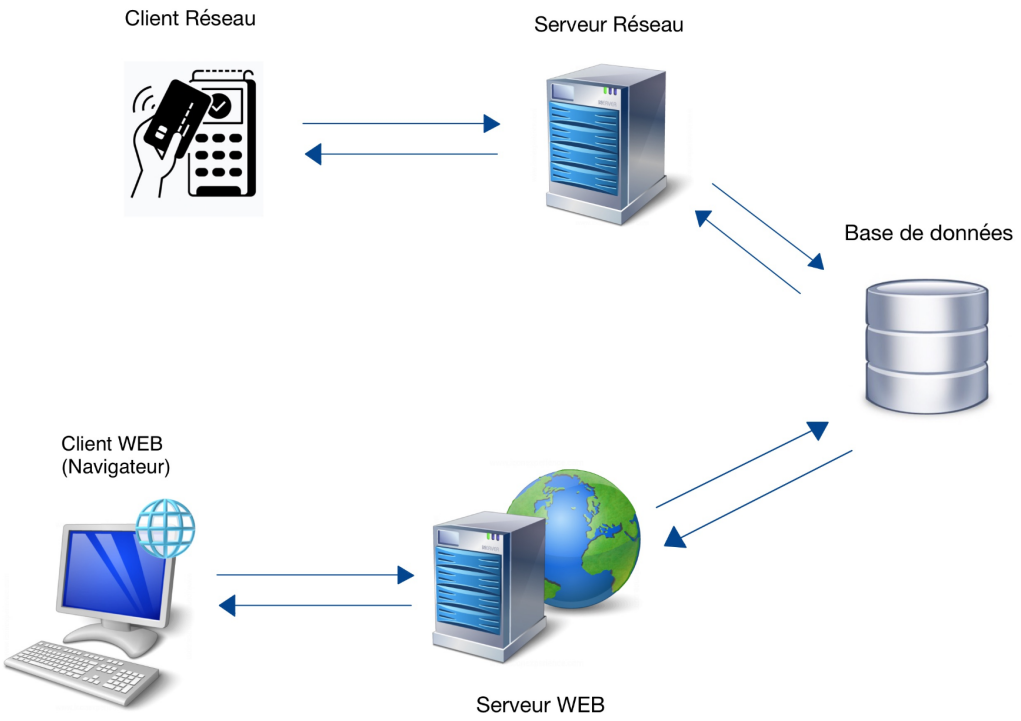
échanges HTTP(S) entre le serveur WEB et le Navigateur :

- **Navigateur** : est un logiciel conçu pour consulter et afficher le World Wide Web. Il représente dans notre projet le client du serveur Web.
- **Serveur web:** est un logiciel de service de ressources Web. Il répond aux requêtes adressées par le navigateur

Les deux niveaux sont liées par la base de données, ainsi le serveur WEB et le serveur

réseau communiquent tous les deux avec la base de données :

- **Base de données:** permet de stocker et de retrouver des données structurées, semi-structurées ou des données brutes ou de l'informations



Dans ce projet on utilise des notions de base déjà définies, à savoir :

Personne : de manière générale définit une personne avec toutes ses caractéristiques (nom, prénom, sexe, adresse mail….)

Membre: une personne qui, grâce à un abonnement peut accéder à la salle ainsi qu'aux différents équipements de la salle, et au parking

Employé: une personne qui à rôle précis à jouer dans la salle (coach, accueil, gardien)

Cours: représente la liste des cours disponibles dans la salle (yoga, musculation, cardio, boxe..) avec une date, une heure de début et une heure de fin

Carte: la carte membre ou employé qui va permettre aux personnes de rentrer dans la salle dans les zones qui leur sont réservées

Equipement: référence les différents équipements disponibles dans les différentes salles, ainsi que leurs disponibilitées

Portique: équipement qui va permettre ou non d'accéder à la salle de sport en scannant sa carte, et à partir du type de l'abonnement aura accès ou non à la zone

Parking: partie de la salle de sport, où une personne pourra choisir ou pas de réserver une place pour sa voiture, il est défini par son adresse et de sa capacité maximale

Véhicule : le véhicule que la personne aura choisi de garer dans le parking de la salle, il a un type et un ID unique .

Réservation: va permettre au membre de réserver un cours grâce à l'id du membre

Salle: définit la salle de sport, avec toutes les informations qui lui sont associées tel que l'adresse et le nom de la salle

Scanne : à chaque fois qu'une personne scanne sa carte cette action est enregistrée avec l'id de la porte ainsi que l'id du portique où la personne a scanné

3.2 Fonctionnalités du système

Le système informatique doit permettre de réaliser les opérations suivantes.

Depuis le réseau de la salle de sport :

- Une personne devra être en mesure de scanner sa carte (que la personne soit un membre ou un employé), le système devra lui dire si son abonnement est actif ou pas, ainsi que lui dire s'il se présente au bon portique
- Une personne devra avoir la possibilité de choisir de réserver une place de parking ou pas, si la personne décide de réserver la place sa voiture est enregistrée avec un id unique

Depuis le site Web de la salle de sport nous devons être en mesure de :

- Consulter les différentes informations liées aux différentes salles, tel que la liste des cours disponibles, l'adresse de la salle, ainsi que les différents équipements disponibles
- Un membre doit être en mesure de se connecter sur le site et avoir la possibilité de modifier ses informations personnelles, et doit pouvoir réserver les cours auxquels il souhaite participer .
- Un employé doit aussi avoir la possibilité de se connecter

4. Technicité des échanges réseau entre le client réseau et le serveur réseau :

4.1. Mise en œuvre du protocole entre le serveur réseau et le client réseau

Le protocole utilisé pour les échanges entre le serveur réseau et le client réseau dans notre

projet est le protocole TCP car :

•Il permet d'assurer le transfert des données de façon fiable.

•Il est efficace car il assure la transmission de toutes les données envoyées en toute

sécurité.

•Il permet de multiplexer les données, c'est-à-dire de faire circuler simultanément des

informations provenant de sources (applications par exemple) distinctes sur une même ligne.

•Il permet enfin l'initialisation et la fin d'une communication de manière courtoise.

Le port choisi pour les échanges client-serveur est 65433.

4.2. Programmation des sockets réseau

Dans le cadre du projet nous avons programmé les sockets réseau en Java et en Python.

Le socket du côté client réseau est programmé en JAVA, et le socket du côté serveur réseau
est programmé en Python.

Tout d'abord, nous avons hébergé notre base de données sur alwaysdata, une fois connecté à cette base de données depuis notre serveur nous aurons la possibilité d'effectuer des requêtes SQL qui nous intéressent

.Serveur python (server.py): script python qui va nous servir de serveur et qui va traiter les informations que le client lui envoie

Bibliothèques importées:

def _init_: initialise une connexion à la base de données avec l'utilisation des paramètres fournis, ce qui va nous permettre d'utiliser des requêtes SQL à l'aide du curseur créé.

def_execute: prend une requete SQL, et l'exécute à l'aide du curseur associé à la connexion

def fetchone(self): récupère la prochaine ligne de résultat de la dernière requête exécutée à l'aide du curseur.

def close(self): ferme le curseur et la connexion à la base de données.

del log(message): cette fonction va nous permettre d'avoir les logs et d'avoir une sorte de trace de ce qu'il se passe entre le serveur et le client, avec l'heure à laquelle on a fait le traitement

def verifier_abonnement_actif_by_id_carte(id_carte): cette fonction prend l'ID d'une carte, vérifie l'état de l'abonnement associé, enregistre des logs en conséquence, et retourne un message indiquant si l'abonnement est actif, expiré, invalide, ou s'il y a une erreur lors de la vérification.

def get_type_carte(id_carte): cette fonction permet d'obtenir le type de carte associé à un ID de carte en effectuant une requête SQL. Elle renvoie le type de carte s'il est trouvé et génère un message d'erreur en cas de problème.

def carte_exist(id_carte): cette fonction permet de vérifier si une carte avec l'ID spécifié existe dans la base de données. Elle retourne "TRUE" si la carte existe et "FALSE" sinon, en générant un message d'erreur en cas de problème.

def verif_accees_portique(id_portique): cette fonction vérifie l'accès à un portique en fonction de l'ID du portique, en comparant le type de carte de l'utilisateur avec le type spécifié pour le portique. Elle enregistre des logs en fonction des résultats de la vérification et retourne un message indiquant si l'accès est autorisé ou non.

def verif_est_un_membre(idc): cette fonction permet de vérifier si une carte est associée à un membre en récupérant et retournant le type de carte à partir de la base de données. Elle génère un message d'erreur en cas de problème lors de la vérification de la carte.

def verif_abo_premium(idc): cette fonction permet de vérifier si un membre associé à une carte à un abonnement premium en fonction du type de carte et du type d'abonnement dans la base de données. Elle retourne des messages indiquant le résultat de la vérification ou une chaîne vide en cas d'autres types de cartes. En cas d'erreur lors de l'exécution des requêtes SQL, elle retourne "NON".

def verif_parkingexist(idparking):cette fonction permet de vérifier si un parking avec l'ID spécifié existe dans la base de données. Elle retourne "PARKING OUI" si le parking existe et "PARKING NON" sinon, en générant un message d'erreur en cas de problème.

def verif_vehi_exist(vehi): vérifie l'existence d'un type de véhicule dans la base de données en fonction du type de véhicule fourni en argument.

def obtenir_dernier_id_vehicule(): vise à obtenir le dernier identifiant de véhicule dans la base de données.

def gener_id_vehi(): génère un nouvel identifiant de véhicule en se basant sur le dernier identifiant existant dans la base de données.

def maj_cap_max(id_parking): cette fonction met à jour la capacité maximale d'un parking en la décrémentant de 1 si la capacité actuelle moins 1 est supérieure ou égale à 0. Elle retourne la capacité maximale mise à jour ou la capacité maximale originale en fonction du résultat de la mise à jour. Elle enregistre également des logs pour indiquer le résultat de l'opération.

def creer_reserv_véhicule(idparking, vehi): cette fonction crée une réservation de véhicule dans un parking spécifié en générant un nouvel identifiant de réservation, l'enregistrant dans la base de données, et enregistrant les résultats dans les logs. Elle retourne un message indiquant que la réservation a été prise en compte.

Libraires utilisées :

- **psycopg2 :** permet la connexion à la base de données
- **socket:** permet l’initialisation et l’utilisation de socket
- **datetime:** permet de récupérer l’heure exacte

.**Client Java (client.java):**

Liste des import que nous avons fait pour le client java :

- import java.io.BufferedReader;
- import java.io.IOException;
- import java.io.InputStreamReader;
- import java.io.PrintWriter;
- import java.net.Socket;

Le client, dans notre projet va donc nous permettre de dialoguer avec le serveur et demander certaines informations

5. Dictionnaire de données

Ici, nous allons présenter notre dictionnaire de données qui va nous servir pour la concrétisation de notre projet :

nom attribut	type	null / not null	domaine	remarques
Personne				
id_personne	VARCHAR(9)	NN	alphanumérique	PK
nom	VARCHAR(30)	NN	alphabétique	
prénom	VARCHAR(30)	NN	alphabétique	
sexe	CHAR(1)	NN	F/M	
num_téléphone	CHAR(10)	NN	numerique	
adress_mail	VARCHAR(30)	NN	alphabétique	
login	VARCHAR(20)	NN	alphanumérique	
mot_de_passe	VARCHAR(20)	NN	alphanumérique	
Membre				
id_membre	VARCHAR(9)	NN	alphanumérique	PK
type_abonnement	VARCHAR(20)	NN	Classique/premium/famille	CONSTRAINT
Employé				
id_employé	VARCHAR(9)	NN	alphanumérique	PK

poste	VARCHAR(20)	NN	gardien/accueil/coach	CONSTRAINT
Reservation				
id_reservation	CHAR(9)	NN	alphanumérique	PK
date_début	DATE	NN	01/01/2022<=date_début<=31/12/2023	
date_fin	DATE	N	01/01/2022<=date_fin<=31/12/2023	
Cours				
id_cours	CHAR(8)	NN	alphanumérique	PK
nom_cours	VARCHAR(30)	NN	alphabétique	
heure_début	TIME	NN	8:00<=heure_début<=20:00	
heure_fin	TIME	NN	10:00<=heure_début<=22:00	
type_cours	VARCHAR(20)	NN	Yoga/musculation/cardio…	
Equipement				
id_equipement	CHAR(9)	NN	alphanumérique	PK
nom_equipement	Varchar(30)	NN	alphabétique	
état	ENUM	NN	Occupé/ disponible	
date_achat	DATE	NN	01/01/2022<=date_achat<=31/12/2023	

Portique				
id_portiqe	CHAR(10)	NN	alphanumérique	PK
type_portique	VARCHAR(20)	NN	Employé/Membre/Mixte	CONSTRAINT
adress_portique	VARCHAR(20)	NN	Salle/Parking	CONSTRAINT
état	VARCHAR(20)	NN	Marche/ hors service	
Salle				
id_salle	int	NN	positif	PK
nom_salle	Varchar(30)	NN	alphanumérique	
date_fondation	DATE	NN	01/01/2022<=date_fondation<=31/12/2023	
Parking				
id_parking	CHAR(8)	NN	alphanumérique	PK
capacité	INTEGER	NN	Numérique capacité>= 0	
Véhicule				
id_véhicule	CHAR(7)	NN	alphanumérique	PK
type_véhicule	VARCHAR(20)	NN	voiture/moto	CONSTRAINT
Carte				
id_carte	CHAR(9)	NN	alphanumérique	PK

type_carte	VARCHAR(20)	NN	Employé/Membre	CONSTRAINT
date_achate	DATE	NN	01/01/2022<=date_achate<=31/12/2023	
date_expiration	DATE	NN	01/01/2022<=date_fondation<=31/12/2023	

6. Echanges réseau

Les échanges réseau d'une salle de sport concernant son accessibilité se font entre le client réseau (lecteur de carte), correspondant au portique de la salle et celui du parking, avec le serveur réseau. Lorsqu'une personne scanne sa carte sur un portique réservé aux membres, le lecteur lit les informations contenues dans la carte et renvoie l'information au serveur réseau. Il y a une communication entre une base de données et ce serveur.
En effet le serveur doit récupérer (à partir des informations obtenus via le client réseau) remonter vers la base de données afin d'obtenir des infos supplémentaires (comme l'accès journalier, le type d'abonnement et si le membre à payé pour le mois actuel …) et vérifier s'il est bien en règle.

Il y a donc 3 cas possibles :

1) l'accès est refusé car portique réservé aux Employés

2) refusé car problème de validité (problème de carte ou abonnement invalide)

3) Carte et abonnement OK donc accès validé.

Même principe pour le parking mais avec une contrainte supplémentaire, la capacité du parking. Une personne ayant accès au parking peut être refusée si le parking est plein. Dans le cas du parking nous avons décidé de faire une situation différente à celle du portique SALLE (Accès via un portique). Ici nous traitons la réservation d'une place de parking.

On a 3 cas,

1) La personne fait une réservation, son véhicule est accepté et la capacité actuelle du parking permet le succès de la réservation. La réservation est validée

2) La personne souhaite réserver une place mais avec un véhicule non autorisé (différent d'une voiture ou d'une moto). La réservation n'a pas lieu.

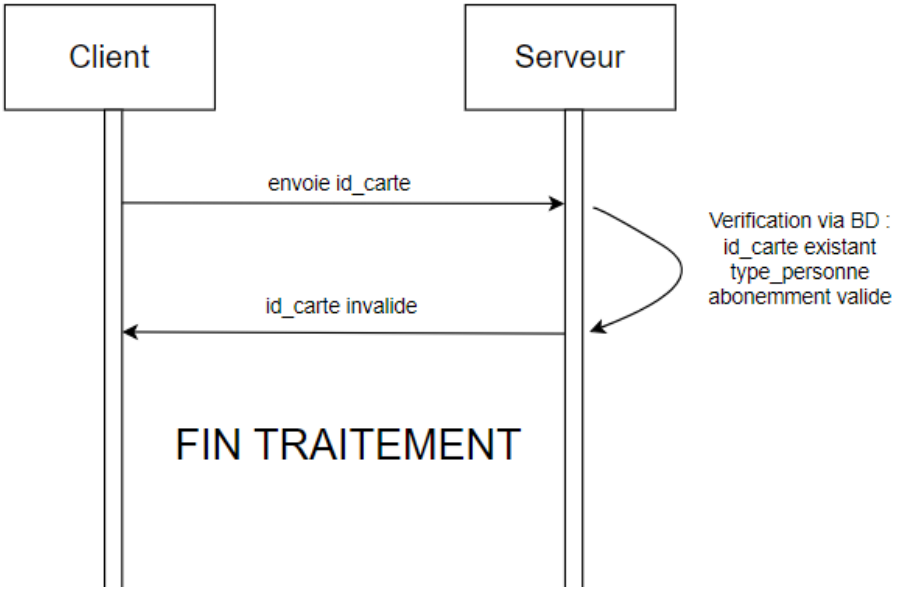
3) La personne souhaite réserver une place mais avec un véhicule autorisé, mais la capacité actuelle du parking est pleine. La réservation n'a pas lieu.

7. Scénarios des échanges réseau

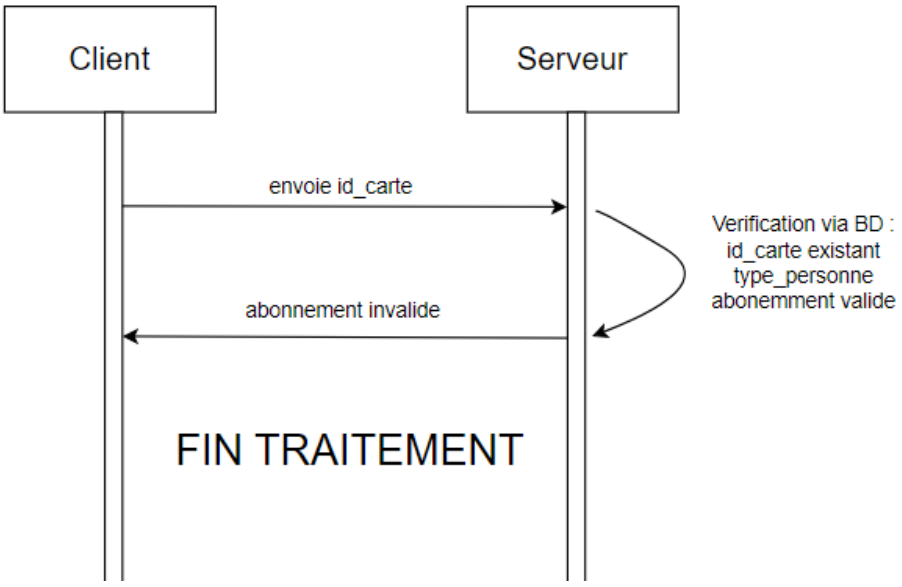
Voici les diagrammes des échanges réseau correspondant aux différents cas possibles.

On s'intéresse en premier temps au portique de la salle de sport.

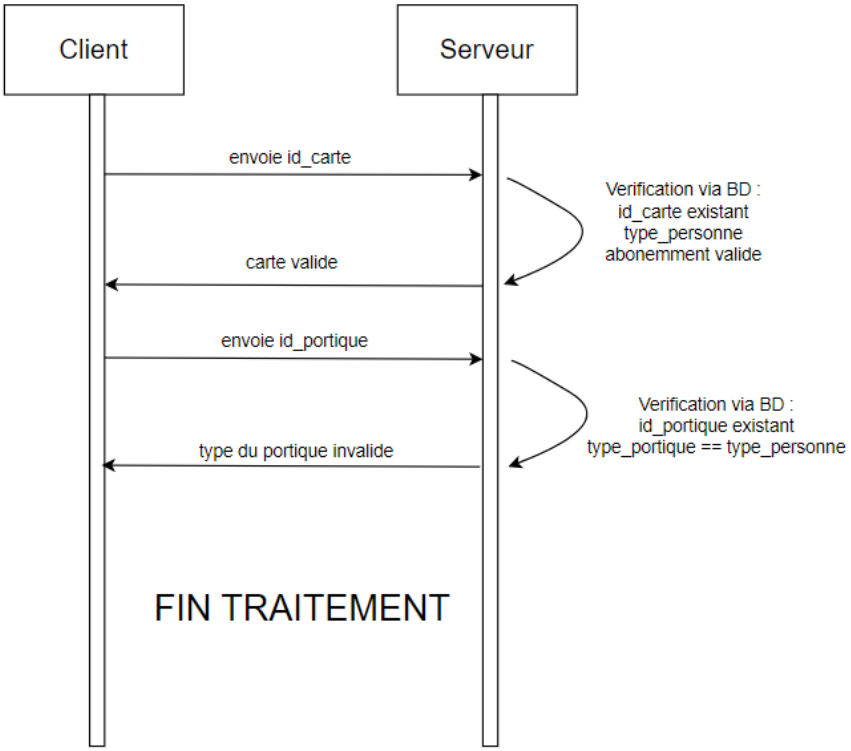
- Cas 1 : Le membre présente une carte (qui n'est plus valide, mais répertoriée dans la base de données). L'accès à la salle de sport est refusé.



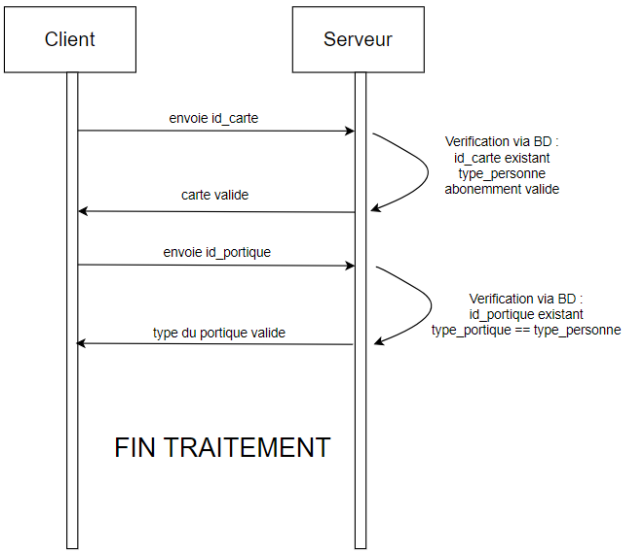
Cas 2 : Le membre présente une carte (répertoriée dans la BD) mais son abonnement a expiré. L'accès à la salle de sport est refusé.



Cas 3 : On part du principe qu'une personne scanne sa carte sur un portique donc que id_portique est TOUJOURS valide. Le membre présente sa carte (valide) sur un portique réservé aux employés. Accès à la salle de sport est refusé



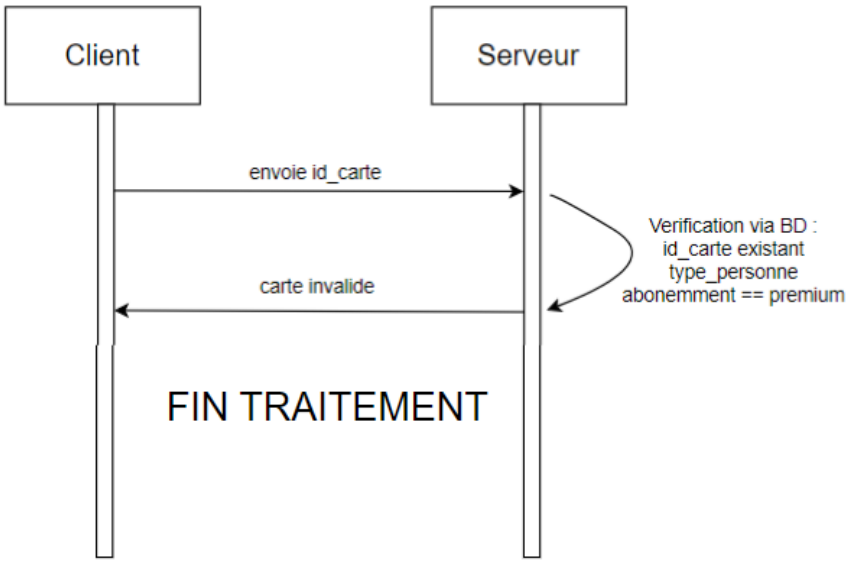
Cas 4 : On part du principe qu'une personne scanne sa carte sur un portique donc que id_portique est TOUJOURS valide. Le membre présente sa carte (valide) sur un portique réservé aux membres. Accès à la salle de sport est acceptée



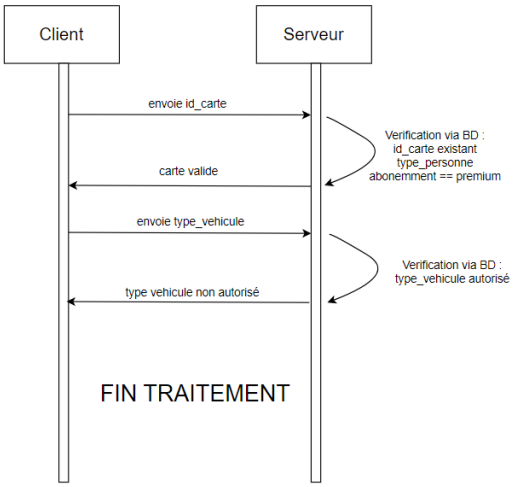
Pour l'accès au parking, nous avons les mêmes échanges réseau que pour le portique SALLE mais sans vérification type_portique car le parking est commun entre les membres et le personnel de la salle de sport. On s'intéresse donc à la réservation d'une place de parking dans une des salles de sport.

Nous avons les mêmes échanges réseau que pour le portiques mais sans vérification type_portique car le parking est commun entre les membres et le personnel de la salle de sport.

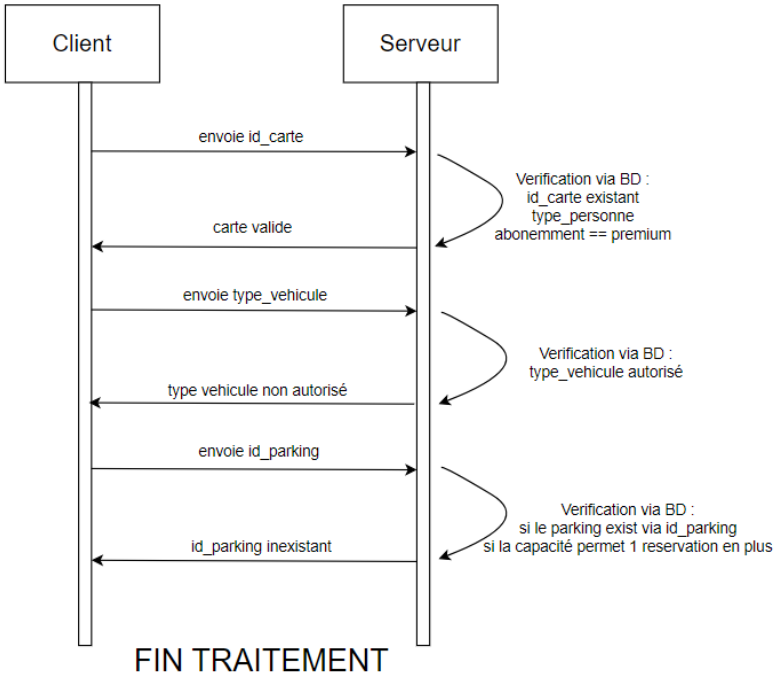
1) La personne souhaite réserver une place mais présente une carte qui est soit inexistante soit son abonnement (dans le cas où la personne est un membre) n'est pas de type "premium". La réservation n'a pas lieu.



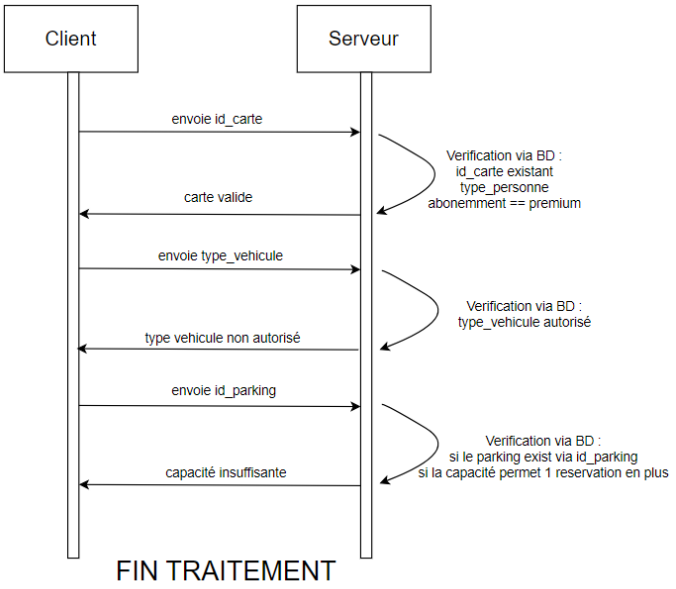
Cas 2 : La personne présente une carte valide, elle est soit un employé soit un membre avec un abonnement premium. Il indique ensuite son type de véhicule mais celui-ci n'est pas accepté. La réservation n'a pas lieu



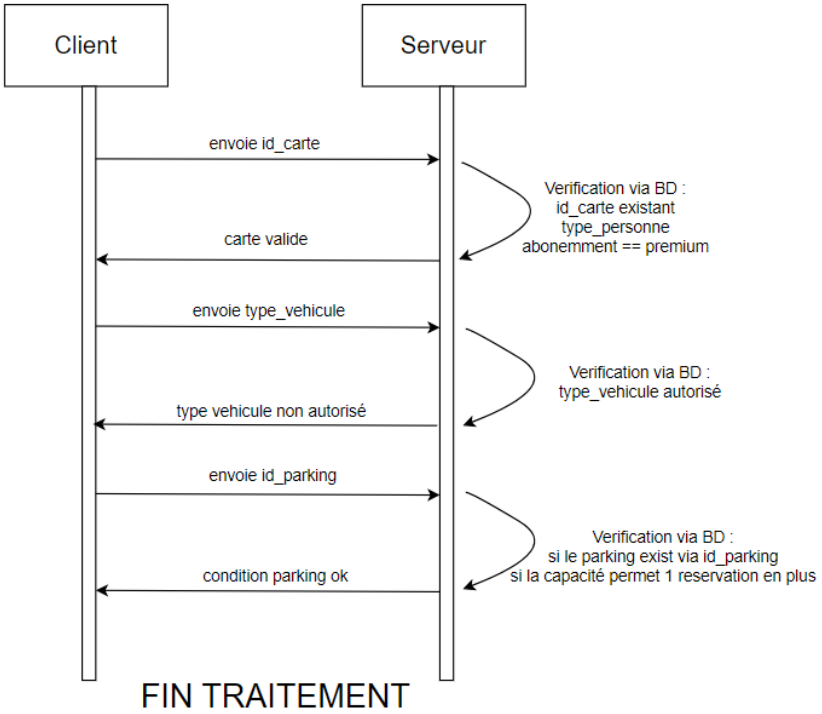
-Cas 3 : La personne présente une carte valide, elle est soit un employé soit un membre avec un abonnement premium. Il indique ensuite son type de véhicule qui est accepté.Il indique ensuite via l'id_parking dans quelle parking d'une des trois salles il veut effectuer sa réservation. Cependant l'id parking est incorrect. La réservation n'a pas lieu



-Cas 4 : La personne présente une carte valide, elle est soit un employé soit un membre avec un abonnement premium. Il indique ensuite son type de véhicule qui est accepté.Il indique ensuite via l'id_parking dans quelle parking d'une des trois salles il veut effectuer sa réservation. Cependant le parking a une capacité restante insuffisante pour effectuer la réservation. La réservation n'a pas lieu.



-Cas 5 : La personne présente une carte valide, elle est soit un employé soit un membre avec un abonnement premium. Il indique ensuite son type de véhicule qui est accepté.Il indique ensuite via l'id_parking dans quelle parking d'une des trois salles il veut effectuer sa réservation. Le parking a une capacité suffisante. La réservation a lieu.



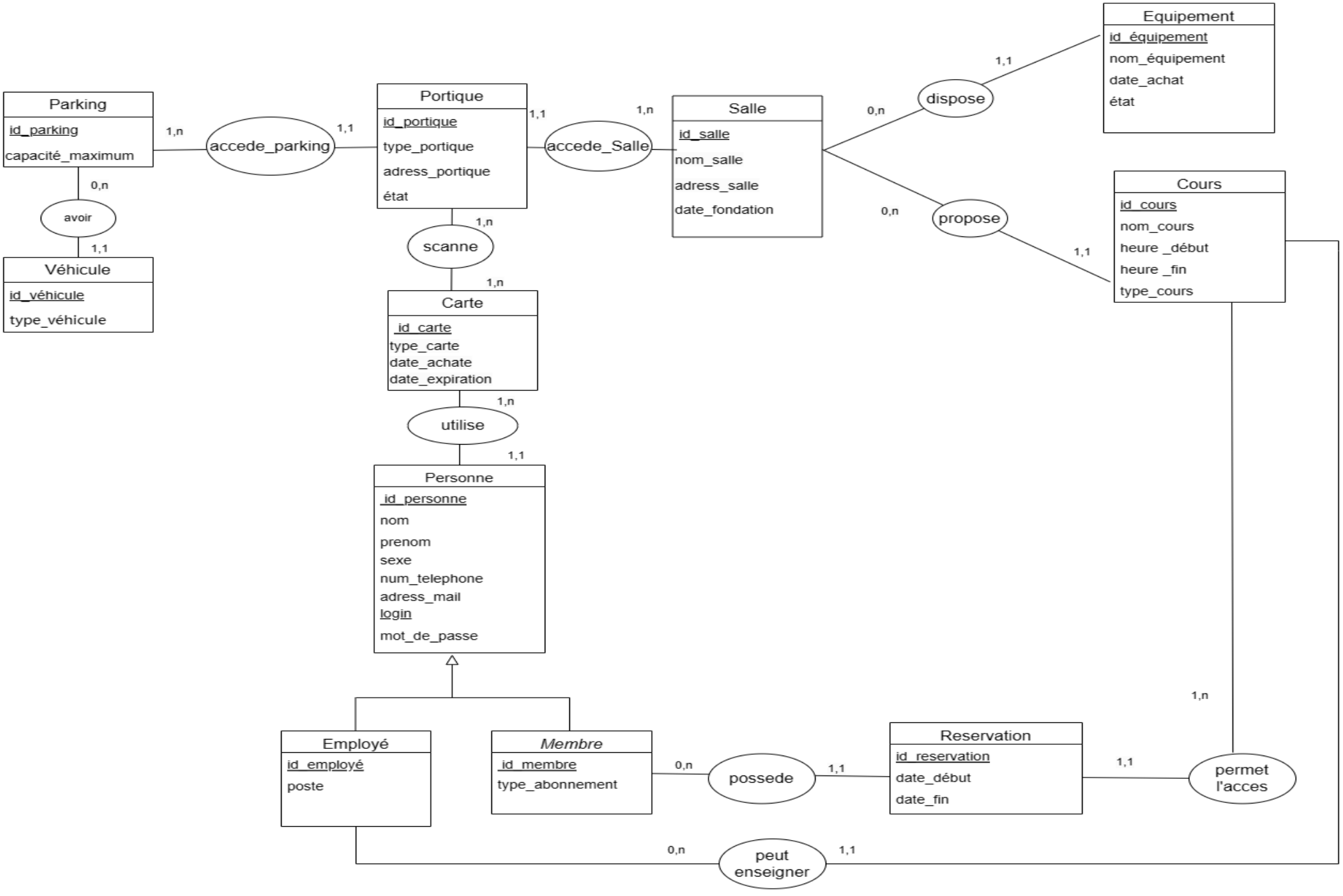

Parking
id_parking
capacité_maximum
1,n
accede_parking
1,1
Portique
id_portique
type_portique
adress_portique
état
1,1
accede_Salle
1,n
Salle
id_salle
nom_salle
adress_salle
date_fondation
0,n
dispose
1,1
Equipement
id_équipement
nom_équipement
date_achat
état
0,n
avoir
1,1
Véhicule
id_véhicule
type_véhicule
1,n
scanne
1,n
Carte
id_carte
type_carte
date_achate
date_expiration
1,n
utilise
1,1
Personne
id_personne
nom
prenom
sexe
num_telephone
adress_mail
login
mot_de_passe
0,n
propose
1,1
Cours
id_cours
nom_cours
heure _début
heure _fin
type_cours
Employé
id_employé
poste
Membre
id_membre
type_abonnement
0,n
possede
1,1
Reservation
id_reservation
date_début
date_fin
1,1
permet l'acces
1,n
0,n
peut enseigner
1,1

Schéma Relationnel

Membre(#id_membre, type_abonnement)

Employé(#id_employé, poste)

Personne(id_personne, nom, prénom, sexe, num_téléphone, adress_mail,login, mot_de _passe, #id_carte)

Portique(id_portique,type_portique,adress_portique,état,#id_parking,#id_salle)

Carte(id_carte,type_carte,date_achate,date_expiration)

Scanne(#id_portique,#id_carte)

Reservation(id_reservation,date_début, date_fin, #id_membre,#id_cours)

Cours(id_cours, nom_cours, heure_début, heure_fin,date_cours, #id_salle,#id_employé)

Équipement(id_équipement,nom_équipement, date_achat, état, #id_salle)

Salle(id_salle,nom_salle, date_fondation,adress_salle)

Parking(id_parking, capacité)

Véhicule(id_véhicule, type_véhicule, #id_parking)

8. Jeu de données

Dans cette section, nous allons présenter notre jeu de données, des exemples concrets des tables que nous allons créer et utiliser pour notre projet :

Table carte

```
bouhadoun_projet=> SELECT * FROM carte;
  id_carte   | type_carte | date_achat | date_expiration
-------------+------------+------------+-----------------
 carteA001   | Membre     | 2022-01-01 | 2023-01-01
 carteA002   | Membre     | 2022-03-01 | 2023-03-01
 carteA003   | Membre     | 2022-05-01 | 2022-05-01
 carteA004   | Membre     | 2023-11-21 | 2024-11-21
 carteA005   | Membre     | 2023-11-21 | 2024-11-21
 carteA006   | Membre     | 2023-11-22 | 2024-11-22
 carteA007   | Membre     | 2023-11-23 | 2024-11-23
 carteA008   | Membre     | 2023-11-23 | 2024-11-23
 carteA009   | Membre     | 2023-11-23 | 2024-11-23
 carteA0010  | Membre     | 2023-11-23 | 2024-11-23
 carteA0011  | Membre     | 2023-11-23 | 2024-11-23
 carteA0012  | Membre     | 2023-11-23 | 2024-11-23
 carteB001   | EmployÚ    | 2022-07-01 | 2030-01-01
 carteB002   | EmployÚ    | 2022-09-01 | 2030-01-01
 carteC001   | EmployÚ    | 2022-09-01 | 2030-01-01
 carteD001   | EmployÚ    | 2022-09-01 | 2030-01-01
 carteA0013  | Membre     | 2023-11-26 | 2024-11-26
(17 lignes)
```

Table cours:

```
bouhadoun_projet=> SELECT * FROM cours;
 id_cours |   nom_cours   | heure_dÚbut | heure_fin |  type_cours  | id_salle | id_employÚ
----------+---------------+-------------+-----------+--------------+----------+-----------
 cours001 | Yoga          | 08:30:00    | 10:00:00  | Yoga         | salle001 | B1
 cours002 | Musculation   | 10:00:00    | 11:30:00  | musculation  | salle001 | B1
 cours003 | Cardio        | 14:00:00    | 15:30:00  | cardio       | salle001 | B1
 cours004 | Boxe          | 16:00:00    | 17:30:00  | Boxe         | salle002 | B2
 cours005 | Pilates       | 18:00:00    | 20:00:00  | Pilates      | salle002 | B2
 cours006 | Zumba         | 10:00:00    | 12:00:00  | Zumba        | salle003 | B2
 cours007 | Powerlifting  | 10:00:00    | 12:00:00  | Powerlifting | salle003 | B1
 cours008 | Cardio        | 16:00:00    | 17:30:00  | cardio       | salle003 | B1
 cours009 | Cardio        | 14:00:00    | 15:30:00  | cardio       | salle002 | B1
(9 lignes)
```

Table employé:

```
bouhadoun_projet=> SELECT * FROM employe;
 id_employe |  poste
------------+---------
 B1         | coach
 B2         | coach
 C1         | accueil
 D1         | gardien
(4 lignes)
```

Table équipement :

```
bouhadoun_projet=> SELECT * FROM equipement;
 id_equipement |   nom_equipement    |    Útat    | date_achat | id_salle
---------------+---------------------+------------+------------+----------
 equipement001 | Tapis de course     | disponible | 2022-02-01 | salle001
 equipement002 | HaltÞres            | disponible | 2022-03-15 | salle001
 equipement003 | VÚlo elliptique     | disponible | 2022-04-20 | salle001
 equipement004 | Rameur              | disponible | 2022-05-10 | salle002
 equipement005 | Banc de musculation | disponible | 2022-06-05 | salle002
(5 lignes)
```

Table membre:

```
bouhadoun_projet=> SELECT * FROM membre;
 id_membre | type_abonnement
-----------+-----------------
 A13       | Classique
 A1        | Classique
 A3        | famille
 A4        | premium
 A6        | Classique
 A7        | premium
 A9        | famille
 A5        | Classique
 A11       | Classique
 A12       | Classique
(10 lignes)
```

Table parking:

```
bouhadoun_projet=> SELECT * FROM parking;
 id_parking  | capacite
-------------+----------
 parking002  |      246
 parking003  |       95
 parking001  |       62
(3 lignes)
```

Table personne:

```
bouhadoun_projet=> SELECT * FROM personne;
 id_personne |   nom    | prenom  | sexe | num_telephone |     adress_mail     |   login    | mot_de_passe |  id_carte
-------------+----------+---------+------+---------------+---------------------+------------+--------------+------------
 A6          | fry      | zzo     | M    | 1346792580    | F@hotmail.com       | fry        | f1           | carteA006
 A1          | Dupont   | Pierre  | M    | 0123456789    | pierre@gmail.com    | Pierre1234 | Pierre@1#2   | carteA001
 A3          | Martin   | Jean    | M    | 0123456789    | jean@outlook.com    | Jean9999   | Jean@5%6     | carteA003
 B1          | Lefebvre | Sophie  | F    | 9876543210    | sophie@gmail.com    | Sophie8888 | Sophie@7&8   | carteB001
 D1          | Lopez    | Maria   | F    | 9876543210    | maria@yahoo.com     | Maria5678  | Maria@3$4    | carteD001
 A7          | lounis   | bhd     | M    | 4674567467    | GG@gmail.com        | loubhd     | 123GOUBERT   | carteA007
 A9          | Olivia   | Yang    | F    | 0192837465    | Olivia@gmail.com    | Oli1       | O123         | carteA009
 A5          | GOUBERT  | ClÚment | M    | 0644445555    | c@gmail.com         | Clem       | C123         | carteA005
 A4          | go       | sy      | F    | 0102030405    | sy@gmail.com        | sy         | sysy         | carteA004
 A11         | nasro    | rona    | F    | 0660872103    | ronanasro@gmail.com | ronas      | 07041959     | carteA0011
 A12         | gou      | vivi    | F    | 0606060644    | sysy@gmail.com      | gous       | Vivi         | carteA0012
 A13         | chef     | cuisto  | M    | 2211221122    | nj@ho.fr            | aa12       | A123456*     | carteA0013
 B2          | Leclerc  | Luc     | M    | 0123456789    | luc@yahoo.com       | Luc6666    | Luc@9*0      | carteB002
 C1          | Garcia   | Luis    | M    | 0123456789    | luis@gmail.com      | Luis1234   | Luis@1#2     | carteC001
(14 lignes)
```

Table portique:

```
bouhadoun_projet=> SELECT * FROM portique;
 id_portique | type_portique | adress_portique |  Útat   | id_parking  | id_salle
-------------+---------------+-----------------+---------+-------------+----------
 portiqueA1  | Mixte         | Parking         | Marche | parking001  |
 portiqueA2  | Membre        | Salle           | Marche |             | salle001
 portiqueA3  | EmployÚ       | Salle           | Marche |             | salle001
 portiqueB1  | Mixte         | Parking         | Marche | parking002  |
 portiqueB2  | Membre        | Salle           | Marche |             | salle002
 portiqueB3  | EmployÚ       | Salle           | Marche |             | salle002
 portiqueC1  | Mixte         | Parking         | Marche | parking003  |
 portiqueC2  | Membre        | Salle           | Marche |             | salle003
 portiqueC3  | EmployÚ       | Salle           | Marche |             | salle003
(9 lignes)
```

Table reservation :

```
bouhadoun_projet=> SELECT * FROM reservation;
 id_reservation | date_dÚbut |  date_fin  | id_membre | id_cours
----------------+------------+------------+-----------+----------
 reserv1        | 2022-01-02 | 2023-12-31 | A1        | cours001
 reserv3        | 2022-05-07 | 2023-12-31 | A3        | cours003
 reserv4        | 2022-07-09 | 2023-12-31 | A3        | cours003
 reserv5        | 2022-09-11 | 2023-12-31 | A1        | cours005
 reserv10       | 2023-11-23 | 2023-12-31 | A3        | cours002
 reserv11       | 2023-11-23 | 2023-12-31 | A7        | cours004
 reserv12       | 2023-11-23 | 2023-12-31 | A9        | cours001
 reserv13       | 2023-11-23 | 2023-12-31 | A5        | cours004
 reserv14       | 2023-11-23 | 2023-12-31 | A12       | cours001
 reserv15       | 2023-11-27 | 2023-12-31 | A13       | cours005
 reserv16       | 2023-11-27 | 2023-12-31 | A5        | cours006
(11 lignes)
```

Table salle:

```
bouhadoun_projet=> SELECT * FROM salle;
 id_salle | nom_salle | date_fondation |     adress_salle
----------+-----------+----------------+----------------------
 salle001 | Salle A   | 2022-03-15     | 123 Main St, City A
 salle002 | Salle B   | 2022-05-20     | 456 Elm St, Town B
 salle003 | Salle C   | 2022-07-10     | 789 Oak St, Village C
(3 lignes)
```

Table scanne:

```
bouhadoun_projet=> SELECT * FROM scanne;
 id_portique | id_carte
-------------+-----------
 portiqueA1  | carteA001
 portiqueA2  | carteB001
 portiqueB2  | carteA002
 portiqueB2  | carteB002
 portiqueB2  | carteC001
(5 lignes)
```

Table véhicule:

```
bouhadoun_projet=> SELECT * FROM vehicule;
 id_vehicule | type_vehicule | id_parking
-------------+---------------+-------------
 vehicule1   | voiture       | parking001
 vehicule2   | moto          | parking002
 vehicule3   | moto          | parking003
 vehicule4   | voiture       | parking001
 vehicule5   | voiture       | parking002
 vehicule6   | moto          | parking003
 vehicule7   | voiture       | parking001
 vehicule8   | moto          | parking001
 vehicule9   | moto          | parking001
 vehicule10  | voiture       | parking001
 vehicule11  | moto          | parking001
 vehicule12  | voiture       | parking001
 vehicule13  | voiture       | parking001
 vehicule14  | voiture       | parking001
 vehicule15  | moto          | parking001
 vehicule16  | voiture       | parking001
 vehicule17  | moto          | parking001
 vehicule18  | voiture       | parking001
 vehicule19  | moto          | parking001
 vehicule20  | voiture       | parking001
 vehicule21  | voiture       | parking001
 vehicule22  | moto          | parking001
 vehicule23  | moto          | parking001
 vehicule24  | moto          | parking001
 vehicule25  | voiture       | parking001
 vehicule26  | moto          | parking002
 vehicule27  | moto          | parking003
 vehicule28  | voiture       | parking003
 vehicule29  | voiture       | parking001
```

9. Requetes SQL significatives :

Dans cette partie nous allons décrire les requêtes SQL les plus importantes que nous avons utilisé dans notre projet:

1.Indique la durée moyenne de tous les cours dans trois salles.（Les résultats sont conservés jusqu'à la première décimale）

```
SELECT S.id_salle, S.nom_salle, ROUND(AVG(EXTRACT(EPOCH FROM (heure_fin - heure_début)) / 3600), 1)
AS durée_moyenne_heures_arrondie
FROM Cours C
JOIN Salle S ON C.id_salle = S.id_salle
GROUP BY S.id_salle, S.nom_salle
HAVING COUNT(C.id_cours) > 0;
```

	id_salle [PK] character	nom_salle character varying (30)	durée_moyenne_heures_arrondie numeric
1	salle001	Salle A	1.5
2	salle002	Salle B	1.7
3	salle003	Salle C	1.8

2.Calculer le nombre de femmes (F) et d'hommes (M) respectivement dans le tableau des personnes.

```
1  SELECT sexe, COUNT(*) AS count_by_gender
2  FROM Personne
3  GROUP BY sexe;
```

Data Output Messages Notifications

	sexe character	count_by_gender bigint
1	M	8
2	F	6

3.Afficher les membres avec leur type d'abonnement et indiquer 'Premium' pour ceux ayant plus de 5 réservations, sinon 'Classique'.

```sql
SELECT id_membre,
       CASE
           WHEN COUNT(id_reservation) > 5 THEN 'Premium'
           ELSE 'Classique'
       END AS type_abonnement
FROM Reservation
GROUP BY id_membre;
```

Data Output | Messages | Notifications

	id_membre character varying (9)	type_abonnement text
1	A7	Classique
2	A5	Classique
3	A13	Classique
4	A3	Classique
5	A9	Classique
6	A12	Classique
7	A1	Classique

4.Le tableau des parkings (Parking) et le tableau des véhicules (Véhicule) sont liés et le nombre de voitures et de motos dans chaque parking est calculé en fonction du groupe de parkings.

```sql
SELECT Parking.id_parking,
    COUNT(CASE WHEN Vehicule.type_vehicule = 'voiture' THEN 1 END) AS voiture_count,
    COUNT(CASE WHEN Vehicule.type_vehicule = 'moto' THEN 1 END) AS moto_count
FROM Parking , Vehicule
WHERE Parking.id_parking = Vehicule.id_parking
GROUP BY Parking.id_parking;
```

Data Output | Messages | Notifications

	id_parking [PK] character	voiture_count bigint	moto_count bigint
1	parking003	2	5
2	parking002	2	3
3	parking001	15	9

5.Interroger tous les équipements de la salle001 et les classer par date_achat du plus récent au plus récent.

```
1  SELECT id_equipement, nom_equipement, état, date_achat
2  FROM Equipement
3  WHERE id_salle = 'salle001'
4  ORDER BY date_achat DESC;
```

Data Output | Messages | Notifications

	id_equipement [PK] character	nom_equipement character varying (30)	état character varying (20)	date_achat date
1	equipement003	Vélo elliptique	disponible	2022-04-20
2	equipement002	Haltères	disponible	2022-03-15
3	equipement001	Tapis de course	disponible	2022-02-01

6.Trouver le même cours dans les trois salles:

```
1  SELECT nom_cours, COUNT(*) AS occurrences
2  FROM Cours
3  GROUP BY nom_cours
4  HAVING COUNT(*) > 1;
```

Data Output | Messages | Notifications

	nom_cours character varying (30)	occurrences bigint
1	Cardio	3

7.Recherchez tous les cours dispensés par des employés ayant le poste de "coach" et triez-les par ID.

```
SELECT P.id_personne, P.nom, P.prenom, C.nom_cours
FROM Personne P, Cours C
WHERE P.id_personne = C.id_employe
AND P.id_personne IN (
    SELECT id_employe
    FROM Employe
    WHERE poste = 'coach'
)
ORDER BY p.id_personne;
```

	id_personne character varying (9)	nom character varying (30)	prenom character varying (30)	nom_cours character varying (30)
1	B1	Lefebvre	Sophie	Yoga
2	B1	Lefebvre	Sophie	Musculation
3	B1	Lefebvre	Sophie	Cardio
4	B1	Lefebvre	Sophie	Powerlifting
5	B1	Lefebvre	Sophie	Cardio
6	B1	Lefebvre	Sophie	Cardio
7	B2	Leclerc	Luc	Boxe
8	B2	Leclerc	Luc	Pilates
9	B2	Leclerc	Luc	Zumba

8.Indique le nombre d'employés dans chaque salle

```
1  SELECT S.id_salle, COUNT(E.id_employe) AS employe_count
2  FROM Salle S, Cours C, Employe E
3  WHERE S.id_salle = C.id_salle
4  AND C.id_employe = E.id_employe
5  GROUP BY S.id_salle;
```

Data Output Messages Notifications

	id_salle [PK] character	employe_count bigint
1	salle001	3
2	salle002	3
3	salle003	3

9.Obtenir toutes les informations sur un membre sans réservation

```sql
SELECT *
FROM Membre
WHERE id_membre NOT IN (
    SELECT id_membre
    FROM Reservation
);
```

Data Output | Messages | Notifications

	id_membre [PK] character varying (9)	type_abonnement character varying (20)
1	A4	premium
2	A6	Classique
3	A11	Classique

10.Comptage des inscriptions par employé : calculez le nombre total d'inscriptions par employé et le nombre de cours correspondant.

```sql
SELECT E.id_employe, P.nom, P.prenom,
    COUNT(DISTINCT R.id_membre) AS total_reservations,
    COUNT(DISTINCT C.id_cours) AS total_cours
FROM Employe E
JOIN Personne P ON E.id_employe = P.id_personne
JOIN Cours C ON E.id_employe = C.id_employe
JOIN Reservation R ON C.id_cours = R.id_cours
GROUP BY E.id_employe, P.nom, P.prenom;
```

Data Output | Messages | Notifications

	id_employe character varying (9)	nom character varying (30)	prenom character varying (30)	total_reservations bigint	total_cours bigint
1	B1	Lefebvre	Sophie	4	3
2	B2	Leclerc	Luc	4	3

11.Statistiques sur les inscriptions aux cours par employé : indiquez le nom du cours et le nombre d'inscriptions par instructeur.

```
SELECT E.id_employe, P.nom, P.prenom, C.nom_cours, COUNT(R.id_reservation) AS nombre_reservations
FROM Employe E
JOIN Personne P ON E.id_employe = P.id_personne
JOIN Cours C ON E.id_employe = C.id_employe
JOIN Reservation R ON C.id_cours = R.id_cours
WHERE E.poste = 'coach'
GROUP BY E.id_employe, P.nom, P.prenom, C.nom_cours
ORDER BY E.id_employe, COUNT(R.id_reservation) DESC;
```

	id_employe character varying (9)	nom character varying (30)	prenom character varying (30)	nom_cours character varying (30)	nombre_reservations bigint
1	B1	Lefebvre	Sophie	Yoga	3
2	B1	Lefebvre	Sophie	Cardio	2
3	B1	Lefebvre	Sophie	Musculation	1
4	B2	Leclerc	Luc	Boxe	2
5	B2	Leclerc	Luc	Pilates	2
6	B2	Leclerc	Luc	Zumba	1

12.Vérifier les cours les plus populaires : découvrez le nom du cours le plus demandé et le nombre de réservations.

```
1  SELECT C.nom_cours, COUNT(R.id_reservation) AS nombre_reservations
2  FROM Cours C, Reservation R
3  WHERE C.id_cours = R.id_cours
4  GROUP BY C.nom_cours
5  ORDER BY COUNT(R.id_reservation) DESC
6  LIMIT 1;
```

Data Output | Messages | Notifications

	nom_cours character varying (30)	nombre_reservations bigint
1	Yoga	3

10. Conclusion et perspectives

Dans cette section, nous résumons la réalisation du projet et nous présentons également les extensions et améliorations apportées du projet.

10.1. Résumé du travail réalisé

Ce projet de gestion de salle de sport, baptisé "**muscleUP**", a été conçu avec succès pour répondre aux besoins essentiels d'une salle de sport moderne. Nous avons mis en place une base de données complète qui englobe les informations relatives aux membres, aux employés, aux cours proposés, aux équipements

disponibles, aux parkings et aux diverses interactions à travers les portiques d'accès.

La réalisation de ce projet a permis de mettre en œuvre une architecture client-serveur, où le client, programmé en Java, communique avec un serveur en Python. Les échanges réseau entre le client et le serveur, gérés via le protocole TCP, ont été implémentés pour vérifier l'accès des personnes aux différentes zones de la salle de sport, en tenant compte de leur type de carte, de leur abonnement et des réservations éventuelles.

10.2.Perspectives

Concernant les perspectives d'amélioration, voici quelques pistes à explorer :

1.Interface Utilisateur Améliorée : Développer une interface utilisateur conviviale pour les membres et les employés, permettant une navigation intuitive et facilitant les réservations de cours, la gestion des abonnements et des informations personnelles.

2.Intégration de la Gestion des Abonnements : Permettre aux employés d'effectuer la gestion des abonnements, y compris les renouvellements, les mises à niveau et les modifications, directement depuis l'interface du logiciel.

3.Amélioration des fonctionnalités réseau : Optimiser les échanges réseau pour une meilleure réactivité, une sécurité renforcée et une gestion plus efficace des accès.

4.Intégration de la Facturation Automatique : Mettre en place un système de facturation automatique pour les membres, en fonction de leur type d'abonnement et des services utilisés.

5.Analyse de Données et Rapports : Développer des outils d'analyse de données pour générer des rapports statistiques sur l'utilisation des équipements, la fréquentation des cours et les tendances des abonnements.