



Module de Gestion de bases de données

Rapport de projet

Création d'une base de données pour un musée avec mise en place de la gestion des éléments important pour la sécurité des œuvres d'arts

Sommaire :

Remerciements	2
Liste des figures	3
Introduction.....	4
I) La base de données SQL	5
1) Prenons l'exemple du script Type_voie	6
2) Explication des tables de notre base de données	7
3) Ordre d'écriture des fichiers SQL DML.....	10
4) Explication UML création d'un nouveau modèle de ronde.....	11
II) Client Udp – Simulateur bâton de rondier	13
III) Serveur Udp – Explications du programme.....	15
1) Cas de la création d'un modèle de ronde	16
Phase 1 : Départ du chef pour création du modèle ronde.....	16
Phase 2 : Ajout de points de passage au modèle de ronde	17
Phase 3 : Clôture du modèle de ronde.....	18
2) Cas de création d'une instance de ronde.....	19
Phase 1 : Départ du garde pour la création de l'instance ronde.....	19
Phase 2 : Ajout de points de passage à l'instance de ronde	20
Phase 3 : Clôture de l'instance de ronde.....	22
IV) Démonstration de notre programme.....	24
Conclusion	34

Remerciements

Notre équipe souhaite tout d'abord remercier nos professeurs de gestion de base données, Monsieur François PECHEUX et Monsieur Francis BRAS pour nous avoir accompagné tout au long de ce sujet, qui nous a permis d'approfondir les enseignements de cette nouvelle matière primordiale avec l'évolution du monde aujourd'hui vers le big data.

Nous les remercions également pour avoir répondu à nos questions et fournis des informations qui nous ont permis d'avancer dans la réalisation de ce projet.

Liste des figures

Figure 1 : Schéma global de notre projet	4
Figure 2 : Image MySQL-Workbench de notre base de données.....	5
Figure 3 : Exemple de la table Type de voie rempli	6
Figure 4 : Code de lecture du fichier event.txt, de concaténation des valeurs et d'envoi.....	14

Introduction

Le module de gestion de base de données a pour objectif de nous sensibiliser aux bases de données et aux interactions possible avec les systèmes embarqués. Nous avons pu étudier le fonctionnement d'une base de données ainsi que sa réalisation avec un outil graphique comme MySQL-Workbench. Également pour interagir avec notre système un apprentissage du langage SQL a été nécessaire.

Ce document a pour objectif de montrer notre compréhension globale du projet ainsi que la mise en place et le fonctionnement de notre base de données.

Le projet musée se déroule de la façon suivante :

La liste des employés est constituée d'agents et de chefs. Le musée détient pour chaque employés leur adresse et numéro de téléphone. Les chefs ont un grade supérieur aux agents et ont pour rôle de créer différent modèle de ronde pour les rondes de sécurité des agents. Les agents ont pour rôle d'effectuer différentes rondes pour assurer de la présence des tableaux au système.

Chaque tableau est constitué d'un tag RFID qui a la lecture émet un numéro unique d'identification. Pour renvoyer cette information nous simulons l'utilisations de bâton de rondier, qui ont pour rôle de taguer chaque tableaux du musée et de renvoyer une trame spécifique au serveur udp.

Nos trames simulée sont constitué des informations suivantes :

- Une adresse mac pour identifier le bâton utilisée. Les adresse mac unique à chaque bâton ont par exemple la forme suivante : B1:2A:B5:AA:CC:01
- Du numéro UID des tag agent ou chef ou des numéros unique des tags repartis dans le musée, ces numéros ont la forme suivante : 00000001
- De leur l'heure du tag ou l'agent a tagué le tableau. Cela prend la forme suivante : 2020-01-20 15:00:00

Pour réaliser les différentes ronde, il faut au préalable qu'elles existent. Pour cela les chefs vont devoir créer chaque ronde une par une. Nous étudierons cette étapes plus loin dans notre document.

Puis les agents pourrons réaliser chaque modèle qui existe dans la table des modèles de ronde, nous étudierons ce cas également dans ce document.

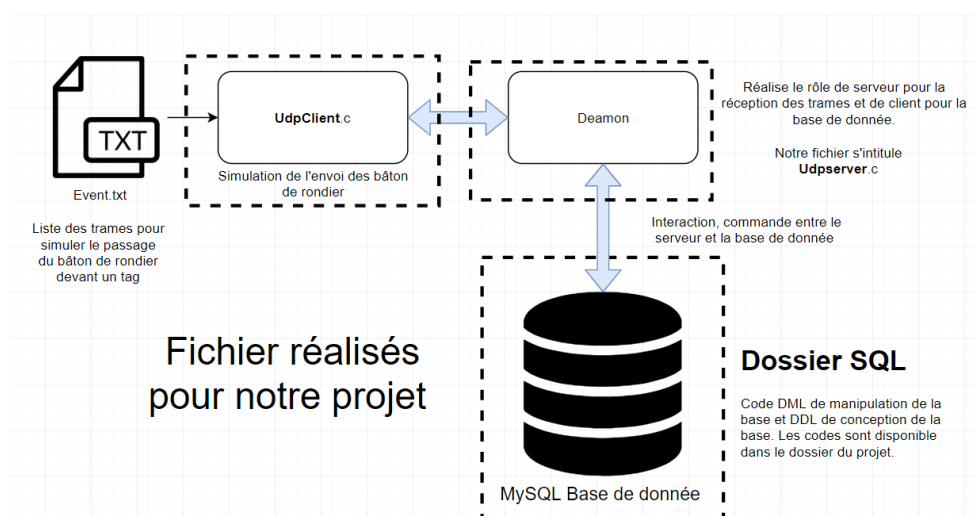


Figure 1 : Schéma global de notre projet

I) La base de données SQL

L'un des objectifs de ce projet est de réaliser une base de données SQL contenant toutes les informations d'un musée. Cette base devra contenir la liste des agent avec comme informations leur adresse, date d'emploi, ect.

Pour ce faire nous avons dessiner notre base de données en utilisant le logiciel MySQL-Workbench dont vous trouverez un extrait en image ci-dessous :

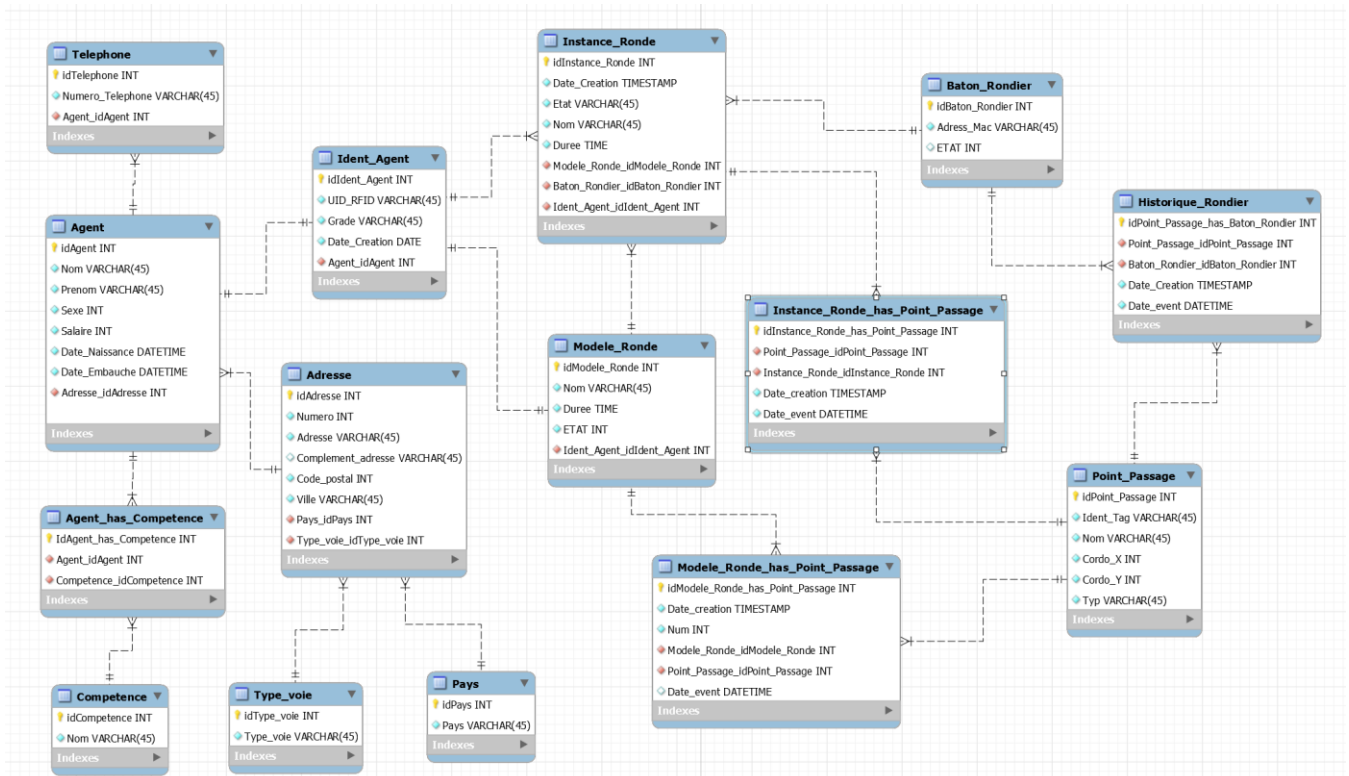


Figure 2 : Image MySQL-Workbench de notre base de données

A partir de cette réalisation nous avons pu exporter un fichier SQL DDL qui permettra sous PhpMyAdmin d'intégrer notre base à la base de données.

Voici la liste des scripts :

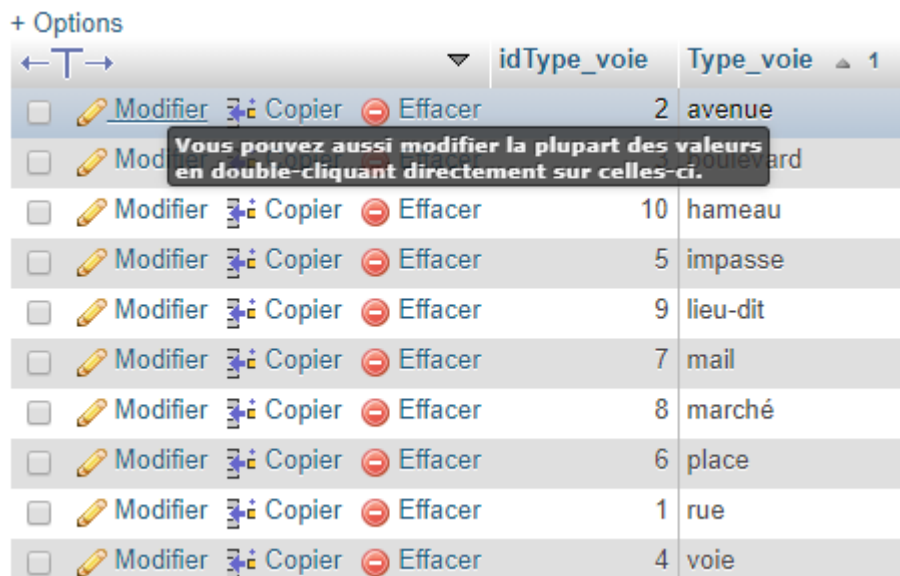
DML	DDL
adresse_DML.sql	rondier_DDL.sql
agent_DML.sql	
agent_has_competence_DML.sql	
baton_rondier.sql	
competences.sql	
identifiant_agent.sql	
Pays_DML.sql	
Points_Passage.sql	
telephone_DML.sql	
Type_voie.sql	

1) Prenons l'exemple du script Type_voie

Pour remplir la table « Type de voies » notre script ressemble à ça :

```
INSERT INTO `Type_voie` (`idType_voie`, `Type_voie`) VALUES ('1', 'rue'),('2', 'avenue'),('3', 'boulevard'),('4', 'voie'),('5', 'impasse'),('6', 'place'),('7', 'mail'),('8', 'marché'),('9', 'lieu-dit'),('10', 'hameau');
```

Ici cela signifie qu'on remplit la table de la manière suivante :



+ Options		idType_voie	Type_voie
<input type="checkbox"/>	Modifier	2	avenue
<input type="checkbox"/>	Modifier	10	hameau
<input type="checkbox"/>	Modifier	5	impasse
<input type="checkbox"/>	Modifier	9	lieu-dit
<input type="checkbox"/>	Modifier	7	mail
<input type="checkbox"/>	Modifier	8	marché
<input type="checkbox"/>	Modifier	6	place
<input type="checkbox"/>	Modifier	1	rue
<input type="checkbox"/>	Modifier	4	voie

Figure 3 : Exemple de la table Type de voie remplie

Pour le reste des tables nous avons réalisé le même type de script INSERT INTO. Pour la table pays vous trouverez dans le dossier un code c qui nous a permis de récupérer une liste de pays d'un fichier.csv. Avec notre programme nous remplissons automatiquement le fichier sql.

Pour remplir toute nos tables comme Agent, point de passage ou bâton de rondier, nous avons suivi les données de monsieur Pêcheux. Cela signifie que les UID RFID et nom de tableaux nous ont été fournis.

Certains scripts utilisent la notion de SELECT du langage SQL pour faire une recherche dans la base de données. Cela nous permet dans certain cas, comme pour le script Adresse d'automatiquement sélectionner les id des tables pays et type de voies. Ainsi nous n'avons pas à réécrire nos scripts si l'une des deux tables change. Nous réalisons à chaque fois une recherche.

Exemple :

```
INSERT INTO `Adresse` (`idAdresse`, `Numero`, `Adresse`, `Complement_adresse`, `Code_postal`,  
`Ville`, `Pays_idPays`, `Type_voie_idType_voie`)
```

```
VALUES (1, '22', 'Des pingouins', NULL, '75012', 'Paris', (SELECT `idPays` from `Pays` WHERE  
`Pays`='France'), (SELECT `idType_voie` from `Type_voie` WHERE `Type_voie`='rue')),
```

```
(2, '44', 'Des clowns', '1er étage', '44000', 'Nantes', (SELECT `idPays` from `Pays` WHERE  
`Pays`='France'), (SELECT `idType_voie` from `Type_voie` WHERE `Type_voie`='avenue'))
```

```
, (3, '33', 'Des artistes', NULL, '28000', 'Chartres', (SELECT `idPays` from `Pays` WHERE  
`Pays`='France'), (SELECT `idType_voie` from `Type_voie` WHERE `Type_voie`='lieu-dit')),
```

```
(4, '1', 'Lemercier', NULL, '75013', 'Paris', (SELECT `idPays` from `Pays` WHERE  
`Pays`='France'), (SELECT `idType_voie` from `Type_voie` WHERE `Type_voie`='rue'));
```

Dans l'exemple en rouge, nous recherchons dans la table pays l'id qui correspond au pays France. Si la table est mise à jour et que cette id est modifié nous pouvons en rechargeant le script dans la base resélectionner le bon id, sans avoir à réécrire notre script.

2) Explication des tables de notre base de données

Table 1, Agent :

Cette table contient la liste de tous les agents (chefs et agents) qui travaillent dans le musée. Cette table est constituée de la colonne id qui donne un numéro unique à chaque agent. Les colonnes Nom, Prénom, etc, Date embauche sont des informations uniques pour chaque employé.

Pour finir on retrouve les colonnes Adresse_idAdresse et Ident_Agent_idIdent_Agent qui sont des foreign key vers les tables Adresse et Ident_Agent.

La relation reliant ces foreign Key est une relation 1:N, un agent ne peut avoir qu'une seule adresse, mais une adresse peut appartenir à plusieurs agents.

Table 2, Téléphone :

Cette table contient une colonne id pour identifier chaque numéro. Une colonne numéro de téléphone. Ainsi qu'une foreign key qui relie cette table à la table agent. Du point de vue de la table Téléphone nous avons une relation N:1 avec Agent. Un agent peut disposer de plusieurs numéros de téléphone, mais un numéro de téléphone appartient à un seul agent.

Table 3, Agent_has_compétence :

Cette table est une table de jonction entre la table de toutes les compétences et de la liste des agents. La relation reliant ces deux tables est une relation N:M.

Cette table fonctionne de la manière suivante :

- Un agent peut avoir plusieurs compétences, et donc une compétence peut avoir plusieurs agents.
- On retrouve deux colonnes dans la table de jonction qui correspondent aux id de chaque agents et compétences.

Table 4, Compétence :

La table regroupe la liste de toutes les compétences qui existent pour les agents.

Table 5, Adresse :

La table regroupe les différentes adresses, on retrouve de manière normale une colonne id ainsi que des colonnes d'information sur les adresses. Enfin, deux colonnes foreign key Pays et Type_voie existent, les deux sont mises en place car plusieurs adresses peuvent se retrouver dans le même pays et le même type de voie.

On peut remarquer qu'utiliser ici deux tables regroupant les types de voies et les pays nous permet d'économiser de l'espace en mémoire, ayant l'information écrite qu'une seule fois dans les deux tables 7 et 6.

Tables 6 et 7, Type_voie et Pays :

Les tables regroupent les différents pays qui existent et les types de voie. Pour les deux une colonne id existe.

Table 8, Ident_Agent :

Cette table crée la liaison entre tous les agents et les différentes rondes. Elle est constituée d'une colonne id, une colonne UID_RFID qui correspond à l'UID du badge de l'agent. A chaque agent est attribué une carte RFID unique. Enfin, on retrouve une colonne Grade qui correspond au grade de l'agent.

Le fait de placer UID_RFID et le grade dans une table de « Transition » nous permet d'avoir d'un côté la liste des agents et de l'autre les différentes rondes et position des agents dans le musée.

Table 9, Instance_Ronde :

Cette table organise l'attribution des rondes de chaque agent, elle regroupe une colonne id pour identifier chaque instance de ronde, d'une colonne date_creation qui indique la date de lancement de la ronde par un agent et des colonnes Etat et Nom. Etat indique dans quel état la ronde se trouve (en cours ou terminée) et nom est une indication pour personnaliser chaque instance de ronde.

Également on retrouve trois Foreign Key reliées avec des relations 1:N. Qui donnent accès à la table modèle de ronde (Modele_Ronde_idModele_Ronde), aux informations d'un bâton de rondier qui lui est attribué (Baton_Rondier_idBaton_Rondier) ainsi qu'à l'agent à qui l'instance est attribuée (Ident_Agent_idIdent_Agent).

Table 15, Instance_Ronde_has_point_Passage :

Dans cette table de jonction on stocke les événements et les passages dans le musée des agents. On stocke dans deux colonnes la date de création de la ligne (Date_creation) ainsi que la date à laquelle l'évènement a eu lieu (Date_event). Il y a également deux Foreign Key, une relié aux points de passage (Point_Passage_idPoint_Passage) et une reliée aux instances de ronde (Instance_Ronde_idInstance_Ronde).

Table 10, Baton_Rondier :

Cette table contient une colonne id pour chaque bâton, une colonne adresse mac qui correspond aux différentes adresse mac de chaque bâton de rondier, ainsi qu'une colonne état qui indique l'état dans lequel le bâton se trouve. C'est-à-dire par exemple 1 pour un modèle de ronde, 2 pour une instance ronde et 0 pour libre.

Table 11, Historique_Rondier :

Nous avons ici une table de jonction entre les différents bâtons de rondier et les points de passage. Pour chaque évènement, c'est-à-dire pour chaque passage d'un bâton sur un tableau physiquement, on rajoute une ligne reliant le bâton au tableau. Également, on rajoute un id dans une colonne id pour identifier chaque ligne. Nous utilisons ici une table de jonction avec une relation N:M car pour chaque bâton de rondier il peut y avoir plusieurs points de passage différents et à l'inverse pour chaque point de passage il peut y avoir plusieurs bâtons de rondier.

Table 12, Point_Passage :

Cette table correspond à liste des différents tableaux du musée. On retrouve une colonne id, une colonne Ident_tag avec le numéro RFID de chaque tableau, le nom du tableau ainsi que son type et les coordonnées de position dans le musée.

Table 14, Modèle_Ronde :

Cette table a pour rôle de définir la liste des rondes qui existe. Attention l'ordre de passage des tableaux dans les rondes est défini dans la table 13. Dans notre table on retrouve une colonne id, une colonne nom, durée (durée estimée de la ronde), état qui par exemple indique l'état dans lequel se trouve la ronde, 1 pour prêt et 0 en cours de création, et une Foreign Key qui détermine l'agent à qui le modèle est attribuée (Ident_Agent_idIdent_Agent).

Table 13, Modele_ronde_has_point_passage :

Cette table de jonction relie les différents tableaux aux rondes qui leur correspondent. C'est dans cette table qu'on attribue chaque ronde et l'ordre de passage des tableaux.

On retrouve la colonne id pour chaque ligne, deux colonnes d'information pour chaque passage avec Date_creation, Num et Date_event. Date_event correspond à l'heure et à la date de passage entre chaque tableau (heure réelle reçue par le bâton), Date_creation correspond à la date de création dans la base de données et num est le numéro qui correspond à l'ordre de passage.

Il y a également deux Foreign Key, une relié aux points de passage (Point_Passage_idPoint_Passage) et une reliée aux modèles de ronde (Modele_Ronde_idModele_Ronde).

3) Ordre d'écriture des fichiers SQL DML

- Pour la conception de la base de données nous devons lancer en premier le fichiers RONDIER_DDL.sql. Un fichier DDL permet de créer la base de données, les tables, les index etc.
- Pour remplir notre base de données nous utilisons les fichiers DML avec un fichier pour chaque tableau. On suit l'ordre suivant pour les lancer :
 - (1) Type_voie et Pays ;
 - (2) Adresse ;
 - (3) Competence ;
 - (4) Agent ;
 - (5) Agent_has_competence ;
 - (6) Telephone ;
 - (7) Ident_Agent ;
 - (8) Baton_Rondier ;
 - (9) Point_Passage

4) Explication UML création d'un nouveau modèle de ronde

1 Phase 1 : départ du chef pour création du modèle ronde

Dans cette première phase, nous mettons en place la création de la ronde. La première étape est :

- Nous récupérons dans la base de données le nombre de modèles de ronde déjà existants avec la requête : "SELECT `Nom` FROM `Modele_Ronde` WHERE 1"
- Récupérer le nombre de modèle déjà existant dans la base (cette valeur peut être de zéro si aucun modèle n'existe) nous permet lors de la création de la ronde de lui donner le nom de modèle + (le nombre de modèle existant + 1)

Dans une deuxième étape nous mettons à jour le bâton de rondier en cours d'utilisation, de la table Baton_Rondier en le passant à l'état 1 (création modèle en cours) avec la requête :

```
"UPDATE `Baton_Rondier` SET `ETAT` = '1' WHERE `idBaton_Rondier` like '%d'", numeroBaton
```

Dans une dernière étape nous créons le modèle de ronde dans la table. Pour ce faire nous récupérons et stockons ensuite l'heure reçue par le client au début de sa ronde. Le premier %d de la requête ci-dessous est un compteur qui s'incrémente pour chaque modèle de ronde présents dans la base. Puis nous insérons dans la base de données le nouveau modèle de ronde créé à l'état 0 (en création) et au temps 0 :0 :0, en la rattachant à l'identifiant uid du chef. Voici la requête correspondante :

```
"INSERT INTO `Modele_Ronde` (`Nom`,`Duree`,`ETAT`,`Ident_Agent_idIdent_Agent`)  
VALUES('modele%d','0:0:0','0',(Select `idIdent_Agent` from `Ident_Agent` WHERE `UID_RFID` = '%s'))",  
batonNumeroModele[numeroBaton], uid
```

La ronde vient d'être initialiser dans la table, il manque maintenant l'ajout des points de passage.

2 Phase 2 : Ajout d'un point de passage au modèle de ronde

Dans cette deuxième phase nous ajoutons tous les points de passage au modèle de ronde qui vient d'être initialisé.

Première étape :

Pour se faire nous sélectionnons dans la table Modele_Ronde, la ronde qui est dans l'état 0 (c'est-à-dire en cours de création) et qui correspond au bâton en cours d'utilisation pour la création de la ronde. Nous utilisons pour cela la trame suivante :

```
"SELECT `idModele_Ronde` FROM `Modele_Ronde` WHERE `ETAT` = '0' AND `Nom` = 'modele%d'",  
batonNumeroModele[numeroBaton]
```

Dans cette requête nous sélectionnons l'id du modèle de ronde pour lequel son état est 0 (en création) et qui porte le nom de notre modèle de ronde (associé à notre bâton) en cours de création sur notre serveur.

Deuxième étape :

Nous insérons ensuite dans notre base de données (dans la table de jonction `Modele_Ronde_has_Point_Passage`) une nouvelle entrée associant le point de passage qui vient d'être tagué avec le modèle de ronde en cours de création :

```
"INSERT INTO `Modele_Ronde_has_Point_Passage`(`Modele_Ronde_idModele_Ronde`,  
`Point_Passage_idPoint_Passage`, `Num`, `Date_event` ) VALUES ('%s',(Select `idPoint_Passage` from  
`Point_Passage` where `Ident_Tag`='%s'), '%d','%s''%s')", modeleCreation, uid,  
batonOrdrePassage[numeroBaton],date,time
```

Où dans l'ordre :

- %s correspond à l'ID du modèle de ronde en cours de création
- %s correspond au numéro uid du point de passage tagué
- %d correspond à l'ordre de passage des points de passage dans lequel se trouve le bâton
- %s %s correspondent à la date et l'heure du tag du point de passage

Vous trouverez des précisions dans notre compte rendu sur l'ajout dans l'historique des points de passage. Comme cette fonction ne fait pas réellement partie de la création d'un modèle de ronde elle n'est pas détaillée dans l'UML. Cependant elle existe et est expliquée dans le compte rendu !

3 Phase 3 : Clôture du modèle de ronde

Avec tous les points de passage insérés dans la table de jonction nous sommes capables de dire pour n'importe quelle ronde qui existe, quelles sont les points de passage qui lui correspondent et dans qu'elle ordre ils doivent être lu.

Pour terminer la création du modèle de ronde le chef doit taguer une dernière fois son badge avec le bâton de rondier.

Première étape :

On refait une vérification avec l'uid pour voir si celui-ci appartient à un chef. Si tel est le cas, nous vérifions ensuite si le modèle de ronde lié au bâton utilisé est bien dans l'état 1 (en cours de création). Si tel est le cas nous stockons l'heure de fin de notre modèle de ronde et on la soustrait à notre heure du début de ronde qui avait été elle aussi stockée. Nous obtenons donc par un calcul la durée de la ronde effectuée.

Nous mettons à jours le modèle de ronde en cours de création avec la durée totale de la ronde et nous mettons à jours sont état :

```
"UPDATE `Modele_Ronde` SET `Duree` = '%s', `ETAT` = '1' WHERE `Nom` like 'modeled'", datetime,  
batonNumeroModele[numeroBaton]
```

Deuxième étape :

Puis pour terminer nous mettons à jours le bâton de rondier en cours d'utilisation, de la table `Baton_Rondier` en le passant à l'état 0 (état libre) avec la requête :

```
"UPDATE `Baton_Rondier` SET `ETAT` = '0' WHERE `idBaton_Rondier` like '%d'", numeroBato
```

Le modèle de ronde est maintenant créé et peut être utilisé par un garde.

II) Client Udp – Simulateur bâton de rondier

Notre projet est de base constitué d'un bâton de rondier qui a pour rôle de lire chaque tag RFID du musée et de les envoyer au serveur udp. Or pour simuler ce bâton de rondier nous avons mis en place un document event.txt qui correspond ligne par ligne à chaque tag que l'agent ou le chef ont tagué pendant leur tour. Le client se connecte au serveur udp et lit toutes les lignes du documents en envoyant chaque ligne une par une.

Ce document event.txt a compose de la manière suivante :

B1:2A:B5:AA:CC:01,20000001,2020-01-20 15:00:00,Le chef 1 démarre OK
B1:2A:B5:AA:CC:01,00000001,2020-01-20 15:02:04,Tag tableau 1 OK
B1:2A:B5:AA:CC:01,00000002,2020-01-20 15:03:58,Tag tableau 2 OK
B1:2A:B5:AA:CC:01,00000003,2020-01-20 15:03:58,Tag tableau 3 OK
B1:2A:B5:AA:CC:01,20000001,2020-01-20 15:05:02,Retag chef 1 fin de ronde OK
B1:2A:B5:AA:CC:01,20000002,2020-01-20 15:30:00,Le chef 2 démarre OK
B1:2A:B5:AA:CC:01,00000004,2020-01-20 15:31:41,Tag tableau 4 OK
B1:2A:B5:AA:CC:01,00000003,2020-01-20 15:35:58,Tag tableau 3 OK
B1:2A:B5:AA:CC:01,00000006,2020-01-20 15:36:45,Tag tableau 6 OK
B1:2A:B5:AA:CC:01,00000007,2020-01-20 15:36:58,Tag tableau 7 OK
B1:2A:B5:AA:CC:01,20000002,2020-01-20 15:37:00,Retag 2 chef fin de ronde OK
B1:2A:B5:AA:CC:02,10000001,2020-01-20 16:00:00,L'agent 1 démarre OK avec modele1,1
B1:2A:B5:AA:CC:02,00000001,2020-01-20 16:02:03,Tag tableau 1 OK
B1:2A:B5:AA:CC:02,00000002,2020-01-20 16:03:58,Tag tableau 2 OK
B1:2A:B5:AA:CC:02,00000004,2020-01-20 16:03:58,Tag tableau 4 ERREUR
B1:2A:B5:AA:CC:02,00000003,2020-01-20 16:05:58,Tag tableau 3 OK
B1:2A:B5:AA:CC:02,10000001,2020-01-20 16:06:07,Retag agent 1 fin instance OK
B1:2A:B5:AA:CC:03,10000002,2020-01-20 16:08:00,L'agent 2 démarre OK avec modele2,2
B1:2A:B5:AA:CC:03,00000004,2020-01-20 16:09:03,Tag tableau 4 OK
B1:2A:B5:AA:CC:03,00000002,2020-01-20 16:10:58,Tag tableau 2 ERREUR
B1:2A:B5:AA:CC:03,00000003,2020-01-20 16:12:58,Tag tableau 3 OK
B1:2A:B5:AA:CC:03,10000002,2020-01-20 16:13:07,Retag agent 2 fin instance ERREUR
B1:2A:B5:AA:CC:03,00000006,2020-01-20 16:14:02,Tag tableau 6 OK
B1:2A:B5:AA:CC:03,00000007,2020-01-20 16:14:45,Tag tableau 7 OK
B1:2A:B5:AA:CC:03,00000007,2020-01-20 16:14:45,Tag tableau 7 ERREUR
B1:2A:B5:AA:CC:03,10000002,2020-01-20 16:15:07,Retag agent 2 fin instance OK

- On retrouve comme première étape la créations par le chef 1 d'un premier modèle de ronde.
- On retrouve en deuxième étape la création d'un second modèle de ronde par le chef 2.
- L'agent 1 réalise le premier modèle de ronde du chef 1. Nous obtiendrons une erreur avec le tableau 4. Le serveur udp vérifie à chaque tableau si l'ordre de suivi de la ronde est le bon, ici ce n'est pas le cas une erreur apparait alors sur le terminal du serveur udp.
- L'agent 2 réalise la deuxième ronde créer pas le chef 2, cette fois nous obtiendrons 3 erreurs de test.

/ !\ Ces cas d'erreur sont développés dans la partie démonstration du programme./ !\

Le programme de base UdpClient.c a été modifié en rajoutant les lignes de code suivantes dans la fonction readFileAdnSendEvent :

```
while (!feof(fic))                                     //On realise la lecture du fichier en boucle tant que on a pas lu toutes les lignes
{
    printf("Appuyez sur une touche pour envoyer une trame\n"); //Il faut appuyer sur une touche pour envoyer une trame
    getchar(); //cela vous permet d'aller dans la base checker que tout fonctionne correctement

    ligne[strlen(ligne)-1]='\0'; //On recupere respectivement chaque valeur en prenant en compte
    char *amac=strtok(ligne,","); //les separateurs
    char *uid=strtok(NULL,",");
    char *datetime=strtok(NULL,",");
    char *comments=strtok(NULL,",");
    char *num_modele=strtok(NULL,",");

    printf("amac=%s| uid=%s| datetime=%s| comments=%s| num_modele=%s|\n",amac, uid, datetime, comments, num_modele); //Affichage des valeurs re

    sprintf(ligne, "%s %s %s,%s,%s", amac, uid, datetime, comments, num_modele); //Concatenation pour respecter la reception des donnees sur l'UDP
    printf("Trame envoyee : %s \n", ligne); //Check et affichage de la trame a envoyer
    sendDatagram(hostname, portno, ligne); //On utilise la fonction d'envoi pour envoyer les trames
    fgets(ligne,100,fic);
}

return 0;
```

Figure 4 : Code de lecture du fichier event.txt, de concaténation des valeurs et d'envoi

- Le getchar() permet de créer une petite temporisation. Cela permet lors du lancement d'envoyer une trame à la fois et d'observer à chaque fois dans phpMyAdmin les bon changement.
- On récupérer en premier toutes les variables de la ligne du documents event.txt. Puis on les concatènes pour respecter l'ordre de réception des trames de notre Udp Serveur.
- On affiche la trame d'envoie type pour vérifier sur le terminal ce qui est envoyé au server avec le printf.
- Dans une trames on trouve en premier l'adresse mac du bâton, puis l'UID RFID de chaque tag, la date et l'heure, le commentaire et potentiellement en plus le choix d'un modèle de ronde pour le début d'une instance de ronde effectuée par un garde.

III) Serveur Udp – Explications du programme

Dans cette partie nous nous proposons de développer en détail le fonctionnement du programme « udpserver.c ». Ce serveur s'occupe de recevoir les données des rondiers (simulés par « udpclient.c ») et communique avec la base de données pour lui envoyer des informations pertinentes.

Afin de pouvoir expliquer au mieux notre programme nous allons ici préciser quelques détails spécifiques à celui-ci. Notre programme est composé de tableaux d'entiers, permettant de stocker les informations relatives à un bâton en cours d'utilisation. Ces variables sont :

```
int numeroBaton = 0;           //Bâton sélectionné (dernier caractère addr mac)
int baton[5];                  //Etat du bâton, type de ronde en cours (0 =
aucune, 1 = modèle de ronde, 2 = instance de ronde)
int batonNumeroModele[5];      //Numéro du modèle en cours
int batonNumeroInstance[5];    //Numéro de l'instance en cours
int batonOrdrePassage[5];      //Ordre des points de passages du bâton
```

Ainsi, au début de notre programme nous déterminons à l'aide d'un Switch Case le numéro du bâton utilisé (1, 2, 3, 4 ou 5). Nous stockons ce numéro dans la variable « numeroBaton ». Ce numéro de bâton de rondier aura donc plusieurs états qui lui sont propres tels que son état d'occupation, le numéro de modèle avec lequel le bâton est exécuté, le numéro de l'instance avec lequel le bâton est exécuté et l'ordre des points de passage où en est le bâton au sein d'un modèle de ronde ou d'une instance de ronde.

Cela permet de distinguer plusieurs bâtons en même temps (jusqu'à 5 dans notre cas) et leur état au sein du programme.

Notre programme est composé de 6 actions possibles :

- Départ création d'un nouveau modèle de ronde.
- Ajout de points de passages à ce modèle de ronde avec ajout dans l'historique rondier.
- Clôture de ce modèle de ronde.
- Départ création d'une instance de ronde en sélectionnant un modèle de ronde.
- Ajout de points de passages à cette instance de ronde avec ajout dans l'historique rondier.
- Clôture de cette instance de ronde.

Nous allons ici détailler ces différentes actions possibles gérées par notre serveur.

Notre programme avant de gérer ces différents événements va consulter la base de données pour vérifier qu'au moins un modèle de ronde existe, avec la requête :

```
"SELECT `Nom` FROM `Modele_Ronde` WHERE 1"
```

Si au moins un modèle de ronde existe dans la base c'est que celle-ci a déjà géré au moins une fois la création d'un modèle de ronde. Nous vérifions donc qu'aucun des bâtons de rondier n'est dans l'état 1 ou 2 (1= en cours de création d'un modèle de ronde et 2= en cours de création d'une instance de ronde) avec la requête :

```
"SELECT `ETAT` FROM `Baton_Rondier` WHERE `ETAT` = '1' OR `ETAT` = '2' AND
`Adress_Mac` ='B1:2A:B5:AA:CC:0%d'", numeroBaton
```


Si un bâton de rondier est dans un de ses deux états alors nous affichons un message d'avertissement qui avertit qu'une coupure du serveur a sûrement eu lieu et qu'il faut supprimer de la base, la ronde (instance ou modèle) en cours.

1) Cas de la création d'un modèle de ronde

Phase 1 : Départ du chef pour création du modèle ronde

Pour démarrer une ronde le chef doit taguer sa base avec un bâton de rondier. Afin de vérifier que le tag provient bien d'un chef nous faisons un test de condition : « `if(uid[0] == '2')` ». Puis nous vérifions que le bâton de rondier utilisé est bien à l'état 0 (donc libre) dans le programme. Une fois ces conditions passées nous récupérons dans la base de données le nombre de modèles de ronde déjà existants avec la requête :

```
"SELECT `Nom` FROM `Modele_Ronde` WHERE 1"
```

Nous mettons ensuite à jour le bâton de rondier en cours d'utilisation, de la table `Baton_Rondier` en le passant à l'état 1 (création modèle en cours) avec la requête :

```
"UPDATE `Baton_Rondier` SET `ETAT` = '1' WHERE `idBaton_Rondier` like '%d'", numeroBaton
```

Nous incrémentons ensuite de 1 la variable « `batonNumeroModele[numeroBaton]` » pour reprendre au modèle `n+1` dans la base de données.

Nous récupérons et stockons ensuite l'heure reçue par le chef au début de sa ronde dans une variable interne de notre programme.

Le premier `%d` de la requête ci-dessous correspond au ième modèle de ronde de la base. Puis nous insérons dans la base de données le nouveau modèle de ronde créé à l'état 0 (en création) et au temps 0 :0 :0, en la rattachant à l'identifiant `uid` du chef qui a tagué. Voici la requête correspondante :

```
"INSERT INTO `Modele_Ronde`  
(`Nom`,`Duree`,`ETAT`,`Ident_Agent_idIdent_Agent`)  
VALUES('modele%d','0:0:0','0',(Select `idIdent_Agent` from `Ident_Agent`  
WHERE `UID_RFID` = '%s'))", batonNumeroModele[numeroBaton],uid
```

La création du départ du modèle de ronde est donc terminée. Le bâton de rondier est occupé et ne peut plus être utilisé par un autre chef ou un garde.

Phase 2 : Ajout de points de passage au modèle de ronde

Afin de créer un modèle de ronde, le chef doit taguer des points de passage. Lorsque le serveur va recevoir l'uid d'un point de passage il pourra le vérifier avec la condition « `if(uid[0] == '0')` ». Une fois cette condition passée nous vérifions l'état du bâton dans notre programme pour vérifier que celui-ci est bien à l'état 1 (1= en cours de création d'un modèle). Puis nous consultons notre base de données afin de récupérer le modèle de ronde en cours d'utilisation avec la requête :

```
"SELECT `idModele_Ronde` FROM `Modele_Ronde` WHERE `ETAT` = '0' AND `Nom` = 'modele%d'", batonNumeroModele[numeroBaton]
```

Dans cette requête nous sélectionnons l'ID du modèle de ronde pour lequel son état est 0 (en création) et qui porte le nom de notre modèle de ronde (associé à notre bâton) en cours de création sur notre serveur.

Nous insérons ensuite dans les points de passage du modèle de ronde une nouvelle entrée associant le point de passage qui vient d'être tagué avec le modèle de ronde en cours de création :

```
"INSERT INTO `Modele_Ronde_has_Point_Passage`(`Modele_Ronde_idModele_Ronde`, `Point_Passage_idPoint_Passage`, `Num`, `Date_event`) VALUES ('%s', (Select `idPoint_Passage` from `Point_Passage` where `Ident_Tag`='%s'), '%d', '%s' '%s')", modeleCreation, uid, batonOrdrePassage[numeroBaton], date, time
```

Où dans l'ordre :

- %s correspond à l'ID du modèle de ronde en cours de création
- %s correspond au numéro uid du point de passage tagué
- %d correspond à l'ordre de passage des points de passage dans lequel se trouve le bâton
- %s %s correspondent à la date et l'heure du tag du point de passage

Nous remplissons ensuite la table historique rondier afin de tracer chaque point de passage tagué.

Pour cela, nous sélectionnons dans un premier temps l'ID du bâton de rondier en cours d'utilisation avec la requête :

```
"SELECT `idBaton_Rondier` FROM `Baton_Rondier` WHERE `idBaton_Rondier` = '%d'", numeroBaton
```

Puis, nous insérons dans la table historique rondier l'ID du bâton de rondier utilisé, l'ID du point de passage tagué et l'heure de l'évènement auquel le point de passage a été tagué :

```
"INSERT INTO `Historique_Rondier`(`Baton_Rondier_idBaton_Rondier`, `Point_Passage_idPoint_Passage`, `Date_event`) VALUES ('%s', (Select `idPoint_Passage` from `Point_Passage` where `Ident_Tag`='%s'), '%s' '%s')", batonModele, uid, date, time
```

Où dans l'ordre :

- %s correspond à l'ID du bâton de rondier
- %s correspond au numéro uid du point de passage tagué
- %s %s correspondent à la date et l'heure du tag du point de passage

Le modèle de ronde a donc grâce à ce système, ses points de passages ajoutés et l'historique de ronde est complété.

Phase 3 : Clôture du modèle de ronde

Pour clôturer le modèle de ronde, le chef doit retaguer sa base. On refait donc une vérification avec l'uid pour voir si celui-ci appartient à un chef. Si tel est le cas, nous vérifions ensuite si le bâton de rondier utilisé est bien à l'état 1 (modèle en cours de création). Si tel est le cas nous stockons l'heure de fin de notre modèle de ronde et on la soustrait à notre heure de début de ronde qui avait été elle aussi stockée. Nous obtenons donc par un calcul la durée de la ronde effectuée.

Nous faisons ensuite une UPDATE de notre base modèle de ronde avec la durée calculée et l'état 1 (disponible) pour le modèle de ronde en cours de création avec la requête :

```
"UPDATE `Modele_Ronde` SET `Duree` = '%s', `ETAT` = '1' WHERE `Nom` like  
'modele%d'", datetime, batonNumeroModele[numeroBaton]
```

Nous mettons ensuite à jour le bâton de rondier en cours d'utilisation, de la table Baton_Rondier en le passant à l'état 0 (état libre) avec la requête :

```
"UPDATE `Baton_Rondier` SET `ETAT` = '0' WHERE `idBaton_Rondier` like  
'%d'", numeroBaton
```

Le modèle de ronde est maintenant créé et peut être utilisé par un garde. Le bâton de rondier peut également de nouveau être utilisé.

2) Cas de création d'une instance de ronde

Phase 1 : Départ du garde pour la création de l'instance ronde

Pour commencer son instance de ronde, le garde doit taguer sa base avec un bâton de rondier. Cela aura pour effet d'envoyer une trame au serveur contenant en plus des informations communes à chaque tag :

- Adresse mac du bâton de rondier,
- UID,
- Date
- Heure

Nous y rajoutons, le modèle de ronde sélectionné. Normalement l'affectation des modèles de ronde aux gardes devrait être géré par un site web. Cela n'ayant pas été demandé nous avons donc simulé cette affectation de modèle en envoyant celui-ci à la fin de la trame lors du début d'instance pour un garde.

Une fois ces différentes informations reçues, nous vérifions que l'uid reçu est bien celui d'un garde avec la condition : « `if(uid[0] == '1')` ». Nous vérifions ensuite que le bâton de rondier utilisé est bien à l'état 0 donc libre.

Puis, dès lors que ces conditions sont vérifiées, nous interrogeons la base avec un SELECT pour vérifier que le modèle de ronde sélectionné par le garde existe bien et est à l'état 1 (Terminé). La requête est la suivante :

```
"SELECT * FROM `Modele_Ronde` WHERE `Nom` = 'modele%d' AND `ETAT` = '1'",selectionModele
```

Si le modèle sélectionné existe nous commençons la création de l'instance. Nous interrogeons donc la base pour savoir combien d'instances sont déjà présentes au sein de la base avec la requête :

```
"SELECT * FROM `Instance_Ronde` WHERE 1"
```

Nous incrémentons ensuite de 1 la variable « `batonNumeroInstance[numeroBaton]` » pour reprendre à l'instance n+1 dans la base de données.

Nous mettons ensuite à jour le bâton de rondier en cours d'utilisation, de la table `Baton_Rondier` en le passant à l'état 2 (création d'une instance en cours) avec la requête :

```
"UPDATE `Baton_Rondier` SET `ETAT` = '2' WHERE `idBaton_Rondier` like '%d'", numeroBaton
```

Nous récupérons et stockons ensuite l'heure reçue par le garde au début de sa ronde dans une variable interne de notre programme. Nous créons ensuite l'instance avec une requête dans laquelle nous précisons l'état de l'instance de ronde, son nom, l'ID du modèle de ronde associé à cette instance, l'ID du bâton de rondier utilisé pour effectuer cette instance et l'ID de l'agent associé à la création de cette instance de ronde. La requête complète de création de l'instance de ronde est la suivante :

```
"INSERT INTO `Instance_Ronde`
(`ETAT`, `Nom`, `Modele_Ronde_idModele_Ronde`, `Baton_Rondier_idBaton_Rondier`,
`Ident_Agent_idIdent_Agent`) VALUES('%s', 'Instance%d', (Select
`idModele_Ronde` from `Modele_Ronde` WHERE `Nom` = 'modele%d'), (Select
`idBaton_Rondier` from `Baton_Rondier` WHERE
`idBaton_Rondier`='%d'), (Select `idIdent_Agent` from `Ident_Agent` WHERE
`UID_RFID` = '%s')) "
,"EN COURS", batonNumeroInstance[numeroBaton], selectionModele,
numeroBaton, uid
```

Où dans l'ordre :

- %s correspond à l'état de l'instance « EN COURS » au début
- %d correspond au numéro d'instance en cours
- %d correspond au numéro du modèle de ronde sélectionné par le garde
- %d correspond au numéro de bâton de rondier en cours d'utilisation
- %s correspond à l'uid du garde qui effectue la ronde

L'instance de ronde est donc créée et le garde peut effectuer ses points de passage en respectant ceux du modèle de ronde. Le bâton de rondier est maintenant occupé et donc inutilisable par un autre garde ou chef.

Phase 2 : Ajout de points de passage à l'instance de ronde

Afin de créer une instance de ronde, le garde doit taguer les points de passage respectant le modèle sélectionné. Lorsque le serveur va recevoir l'uid d'un point de passage il pourra vérifier qu'il s'agit d'un point de passage avec la condition « `if(uid[0] == '0')` ». Une fois cette condition passée nous vérifions l'état du bâton dans notre programme pour vérifier que celui-ci est bien à l'état 2 (2= en cours de création d'une instance).

Dès que ces conditions sont passées nous récupérons l'uid du point de passage attendu à l'étape n de la ronde du garde. La variable qui précise le numéro de l'étape où en est le garde est pour rappel la variable « `batonOrdrePassage[numeroBaton]` » dans le programme. Nous faisons donc un SELECT dans la base de données pour récupérer l'uid à l'étape « `batonOrdrePassage[numeroBaton]+1` » de la ronde du garde. Nous précisons également le nom du modèle associé à l'instance en cours dans la requête. La requête complète pour récupérer le point de passage attendu est la suivante :

```
"SELECT `Ident_Tag` FROM `Point_Passage` WHERE `idPoint_Passage` IN
(SELECT `Point_Passage_idPoint_Passage` FROM
`Modele_Ronde_has_Point_Passage` WHERE `Num` = '%d' AND
`Modele_Ronde_idModele_Ronde`=(SELECT `idModele_Ronde` FROM `Modele_Ronde`
WHERE `Nom` = 'modele%d'))", batonOrdrePassage[numeroBaton]+1,
selectionModele
```

Où dans l'ordre :

- %d correspond à l'étape/numéro où en est le garde dans sa ronde
- %d correspond au numéro du modèle de ronde sélectionné par le garde

Puis nous regardons également le nombre de points de passage que le garde a tagué dans le modèle de ronde de référence sélectionné. Nous stockons cette valeur dans Cela permet de savoir si le garde a terminé son instance de ronde. La requête complète pour récupérer le nombre de points de passage attendus est la suivante :

```
"SELECT `Num` FROM `Modele_Ronde_has_Point_Passage` WHERE  
`Modele_Ronde_idModele_Ronde`= (Select `idModele_Ronde` from `Modele_Ronde`  
where `Nom` = 'modele%d')", selectionModele
```

Ensuite, avec une condition : « **if** (uid[7] == numIdRFID[7] && batonOrdrePassage[numeroBaton] < nbPointsDePassage) », nous vérifions si le dernier caractère de l'uid tagué par le garde (uid[7]) correspond au dernier caractère de l'uid attendu par le modèle de ronde (numIdRFID[7]). Nous vérifions également que le nombre de points de passage tagués par le garde (batonOrdrePassage[numeroBaton]) n'est pas supérieur aux nombres de points de passage du modèle sélectionné (nbPointsDePassage).

Enfin une fois cette condition passée, le point de passage tagué par le garde peut entrer dans la base de données.

Nous insérons donc dans les points de passage de l'instance de ronde une nouvelle entrée associant le point de passage qui vient d'être tagué avec l'instance de ronde en cours de création. Nous y ajoutons également l'heure de tag avec la date « Date_event » correspondant à la date et l'heure de tag réelle par le bâton (utile en cas de perte de réseau). La requête complète pour insérer le point de passage est la suivante :

```
"INSERT INTO  
`Instance_Ronde_has_Point_Passage`(`Instance_Ronde_idInstance_Ronde`,  
`Point_Passage_idPoint_Passage`, `Date_event`) VALUES ((SELECT  
`idInstance_Ronde` FROM `Instance_Ronde` WHERE `Nom` =  
'Instance%d'), (Select `idPoint_Passage` from `Point_Passage` where  
`Ident_Tag`='%s'), '%s' '%s')", batonNumeroInstance[numeroBaton], uid, date,  
time);
```

Où dans l'ordre :

- %d correspond au nom de l'instance de ronde en cours de création
- %s correspond au numéro uid du point de passage tagué
- %s %s correspondent à la date et l'heure du tag du point de passage

Puis nous récupérons dans notre base l'ID du bâton de rondier en cours d'utilisation avec la commande :

```
"SELECT `idBaton_Rondier` FROM `Baton_Rondier` WHERE `idBaton_Rondier` =  
'%d'", numeroBaton
```

Puis, nous insérons dans la table historique rondier l'ID du bâton de rondier utilisé, l'ID du point de passage tagué et l'heure de l'évènement auquel le point de passage a été tagué :

```
"INSERT INTO `Historique_Rondier`(`Baton_Rondier_idBaton_Rondier`,  
`Point_Passage_idPoint_Passage`, `Date_event`) VALUES ('%s', (Select  
`idPoint_Passage` from `Point_Passage` where `Ident_Tag`='%s'), '%s' '%s')",  
batonInstance, uid, date, time
```

Où dans l'ordre :

- %s correspond à l'ID du bâton de rondier
- %s correspond au numéro uid du point de passage tagué
- %s %s correspondent à la date et l'heure du tag du point de passage

L'instance de ronde a donc grâce à ce système, ses points de passages ajoutés et l'historique de ronde est complété.

Phase 3 : Clôture de l'instance de ronde

Pour clôturer l'instance de ronde, le garde doit retaguer sa base. On refait donc une vérification avec l'uid pour voir si celui-ci appartient à un garde. Si tel est le cas, nous vérifions si le bâton de rondier utilisé est bien dans l'état 2 (instance en cours de création).

Puis nous interrogeons notre base de données avec SELECT pour récupérer tous les numéros des points de passages associés au modèle de ronde sélectionné par le garde au début de son instance de ronde. La requête complète pour récupérer tous les numéros des points de passage est la suivante :

```
"SELECT `Num` FROM `Modele_Ronde_has_Point_Passage` WHERE  
`Modele_Ronde_idModele_Ronde`= (Select `idModele_Ronde` from `Modele_Ronde`  
where `Nom` = 'modele%d')", selectionModele
```

Nous regardons ensuite le nombre de « Num » renvoyés par la base de données. Puis dans une condition `if (batonOrdrePassage[numeroBaton]==num_row)` nous regardons si le nombre de points de passage tagués par le garde correspond (`batonOrdrePassage[numeroBaton]`) au nombre de points de passage associés au modèle de ronde sélectionné (`num_row`).

Si tel est le cas nous stockons l'heure de fin de notre instance de ronde et on la soustrait à notre heure de début d'instance de ronde qui avait été elle aussi stockée. Nous obtenons donc par un calcul la durée de la ronde effectuée.

Nous faisons ensuite une UPDATE de notre base instance de ronde pour l'instance en cours de création à laquelle nous y ajoutons sa durée et son état terminé avec la requête :

```
"UPDATE `Instance_Ronde` SET `Duree` = '%s', `Etat` = 'TERMINE' WHERE `Nom`  
like 'Instance%d'", datetime, batonNumeroInstance[numeroBaton]
```

Nous mettons ensuite à jour le bâton de rondier en cours d'utilisation, de la table `Baton_Rondier` en le passant à l'état 0 (état libre) avec la requête :

```
"UPDATE `Baton_Rondier` SET `ETAT` = '0' WHERE `idBaton_Rondier` like  
'%d'", numeroBaton
```

L'instance de ronde du garde est terminée et le bâton de rondier libéré. Nous avons accès à la ronde de l'agent et pouvons voir sa durée et la comparer avec celle du modèle de ronde suivi. Nous pouvons également analyser chacun de ses points de passage avec l'heure Event (heure de tag réelle) et l'heure de création (heure à laquelle l'information est rentrée dans la base). Si un évènement tel qu'un vol se déroulait au sein du musée nous pourrions voir dans les instances de ronde quel garde faisait l'instance de ronde à tel endroit et à telle heure et ainsi déterminer quel est le garde responsable.

IV) Démonstration de notre programme

Afin de pouvoir réaliser la démonstration suivante veuillez suivre le fichier Readme.txt présent à la racine du dossier de rendu.

Voici ci-dessous une capture d'écran de notre fichier events.txt :

```
Bl:2A:B5:AA:CC:01,20000001,2020-01-20 15:00:00,Le chef 1 démarre OK
Bl:2A:B5:AA:CC:01,00000001,2020-01-20 15:02:04,Tag tableau 1 OK
Bl:2A:B5:AA:CC:01,00000002,2020-01-20 15:03:58,Tag tableau 2 OK
Bl:2A:B5:AA:CC:01,00000003,2020-01-20 15:03:58,Tag tableau 3 OK
Bl:2A:B5:AA:CC:01,20000001,2020-01-20 15:05:02,Retag chef 1 fin de ronde OK
Bl:2A:B5:AA:CC:01,20000002,2020-01-20 15:30:00,Le chef 2 démarre OK
Bl:2A:B5:AA:CC:01,00000004,2020-01-20 15:31:41,Tag tableau 4 OK
Bl:2A:B5:AA:CC:01,00000003,2020-01-20 15:35:58,Tag tableau 3 OK
Bl:2A:B5:AA:CC:01,00000006,2020-01-20 15:36:45,Tag tableau 6 OK
Bl:2A:B5:AA:CC:01,00000007,2020-01-20 15:36:58,Tag tableau 7 OK
Bl:2A:B5:AA:CC:01,20000002,2020-01-20 15:37:00,Retag 2 chef fin de ronde OK
Bl:2A:B5:AA:CC:02,10000001,2020-01-20 16:00:00,L'agent 1 démarre OK avec modele1,1
Bl:2A:B5:AA:CC:02,00000001,2020-01-20 16:02:03,Tag tableau 1 OK
Bl:2A:B5:AA:CC:02,00000002,2020-01-20 16:03:58,Tag tableau 2 OK
Bl:2A:B5:AA:CC:02,00000004,2020-01-20 16:03:58,Tag tableau 4 ERREUR
Bl:2A:B5:AA:CC:02,00000003,2020-01-20 16:05:58,Tag tableau 3 OK
Bl:2A:B5:AA:CC:02,10000001,2020-01-20 16:06:07,Retag agent 1 fin instance OK
Bl:2A:B5:AA:CC:03,10000002,2020-01-20 16:08:00,L'agent 2 démarre OK avec modele2,2
Bl:2A:B5:AA:CC:03,00000004,2020-01-20 16:09:03,Tag tableau 4 OK
Bl:2A:B5:AA:CC:03,00000002,2020-01-20 16:10:58,Tag tableau 2 ERREUR
Bl:2A:B5:AA:CC:03,00000003,2020-01-20 16:12:58,Tag tableau 3 OK
Bl:2A:B5:AA:CC:03,10000002,2020-01-20 16:13:07,Retag agent 2 fin instance ERREUR
Bl:2A:B5:AA:CC:03,00000006,2020-01-20 16:14:02,Tag tableau 6 OK
Bl:2A:B5:AA:CC:03,00000007,2020-01-20 16:14:45,Tag tableau 7 OK
Bl:2A:B5:AA:CC:03,00000007,2020-01-20 16:14:45,Tag tableau 7 ERREUR
Bl:2A:B5:AA:CC:03,10000002,2020-01-20 16:15:07,Retag agent 2 fin instance OK
```

La trame du fichier Events envoyé par le **udpclient.c** se définit donc ainsi :

amac uid date time, commentaires, num_modele (sera nulle si rien)

```
sprintf(ligne, "%s %s %s,%s,%s", amac, uid, datetime, commentaires, num_modele);
```

La trame reçue par le client est décomposée ainsi par le **udpserver.c** :

```
amac=strtok(buf, " ");
uid=strtok(NULL, " ");
date=strtok(NULL, " ");
time=strtok(NULL, ",");
commentaires=strtok(NULL, ",");
modeleSelect=strtok(NULL, " ");
```

Nous allons ici détailler les deux parties encadrées en vertes. Soit la création du modèle de ronde 2 par le chef 2 et l'exécution de l'instance 2 par le garde (agent) 2 prenant pour modèle le 2. Dans cette partie nous allons mettre des captures d'écrans des réponses du serveur face aux commandes du client. Toutes les requêtes, erreurs gérées, informations de mises à jour ou de statuts sont gérées pas notre programme et apparaitront sur le terminal du serveur. On peut donc imaginer un fichier log des réponses du serveur en cas d'investigation sur un évènement survenu dans le musée ou simplement pour voir si un garde se trompe souvent de points de passage.

- ✓ Le premier encadré concerne la création d'un modèle (numéro 2) de ronde pour le chef possédant l'uid 20000002. Ce chef démarre donc sa ronde et tague les tableaux :

- Il tague sa base avec le bâton 1 et le serveur confirme la demande du chef :

```
NOUVELLE TRAME RECUE DU CLIENT
Received packet from 127.0.0.1:46055
Data: [B1:2A:B5:AA:CC:01 20000002 2020-01-20 15:30:00,Le chef 2 démarre OK,(null)]

amac=B1:2A:B5:AA:CC:01| uid=|20000002| date=|2020-01-20| time=|15:30:00| comments=|Le chef 2 démarre OK| modeleSelect=|(null)|

PARTIE GESTION DU DEAMON

INFOS DU CLIENT
uid = 20000002
Time = 15:30:00
Date = 2020-01-20
MACadress = B1:2A:B5:AA:CC:01
Modele Selectionne= (null)

requete = SELECT `Nom` FROM `Modele_Ronde` WHERE 1

requete = SELECT `ETAT` FROM `Baton_Rondier` WHERE `ETAT` = '1' OR `ETAT` = '2'
AND `Adress_Mac` ='B1:2A:B5:AA:CC:01'

Information serveur : Il n'y a pas eu de coupure du serveur pendant une creation
de modele ou execution d'une instance de ronde !

Numero du baton utilise par le client = B1:2A:B5:AA:CC:01
Etat du baton (sur le serveur) utilise par le client = 0

requete = SELECT `Nom` FROM `Modele_Ronde` WHERE 1

Information serveur : NOMBRE DE MODELES EXISTANTS DANS LA BASE 1

UN CHEF VIENT DE TAGUER SON BATON A LA BASE POUR COMMENCER SA RONDE !

requete = UPDATE `Baton_Rondier` SET `ETAT` = '1' WHERE `idBaton_Rondier` like '1'

HEURE DEBUT MODELE : 15:30:0

requete=INSERT INTO `Modele_Ronde` (`Nom`,`Duree`,`ETAT`,`Ident_Agent_idIdent_Agent`) VALUES('modele2','0:0:0','0',(Select `idIdent_Agent` from `Ident_Agent` WHERE `UID_RFID` = '20000002'))

Information serveur : Le baton de rondier 1 n'est plus libre
Creation d'un modele de ronde.
```

Etat de la table Modele_Ronde :

	idModele_Ronde	Nom	Duree	ETAT	Ident_Agent_idIdent_Agent
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	modele1	00:05:02	1	1
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	4	modele2	00:00:00	0	2

Etat de la table Baton_Rondier :

	idBaton_Rondier	Adress_Mac	ETAT
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	B1:2A:B5:AA:CC:01	1
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	B1:2A:B5:AA:CC:02	0
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	B1:2A:B5:AA:CC:03	0
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	4	B1:2A:B5:AA:CC:04	0
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	5	B1:2A:B5:AA:CC:05	0

- Il tague le point de passage 00000004 et le serveur confirme la demande du client :

```
NOUVELLE TRAME RECUE DU CLIENT
Received packet from 127.0.0.1:59528
Data: [B1:2A:B5:AA:CC:01 00000004 2020-01-20 15:31:41,Tag tableau 4 OK,(null)]

amac=|B1:2A:B5:AA:CC:01| uid=|00000004| date=|2020-01-20| time=|15:31:41| comments=|Tag
tableau 4 OK| modeleSelect=|(null)|

PARTIE GESTION DU DEAMON

INFOS DU CLIENT
uid = 00000004
Time = 15:31:41
Date = 2020-01-20
MACadress = B1:2A:B5:AA:CC:01
Modele Selectionne= (null)

requete = SELECT `Nom` FROM `Modele_Ronde` WHERE 1

Numero du baton utilise par le client = B1:2A:B5:AA:CC:01
Etat du baton (sur le serveur) utilise par le client = 1

DEMANDE DE TAG DE POINT DE PASSAGE PAR UN CHEF

Information serveur : Le chef en est a son point de passage numero 1

requete = SELECT `idModele_Ronde` FROM `Modele_Ronde` WHERE `ETAT` = '0' AND `Nom` = 'mo
dele2'

requete=INSERT INTO `Modele_Ronde_has_Point_Passage`(`Modele_Ronde_idModele_Ronde`, `Poi
nt_Passage_idPoint_Passage`, `Num`, `Date_event` ) VALUES ('4',(Select `idPoint_Passage`
from `Point_Passage` where `Ident_Tag`='00000004'), '1','2020-01-20'15:31:41')

Information serveur : Le point de passage 00000004 a ete tague pour le modele de ronde

requete = SELECT `idBaton_Rondier` FROM `Baton_Rondier` WHERE `idBaton_Rondier` = '1'

requete=INSERT INTO `Historique_Rondier`(`Baton_Rondier_idBaton_Rondier`, `Point_Passage
_idPoint_Passage`, `Date_event`) VALUES ('1',(Select `idPoint_Passage` from `Point_Passag
e` where `Ident_Tag`='00000004'),'2020-01-20'15:31:41')

Information serveur : VOUS ETES TRACE l'historique du rondier a ete complete avec l'iden
tifianf du baton de rondier utilise pour ce modele de ronde !
```

- Même réponse du serveur pour le tag des points de passage 00000003, 00000006 et 00000007 :

Etat de la table Historique_Rondier :

	idPoint_Passage_has_Baton_Rondier	Point_Passage_idPoint_Passage	Baton_Rondier_idBaton_Rondier	Date_Creation	Date_event
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	15	2	1	2020-01-18 18:01:02	2020-01-20 15:02:04
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	16	3	1	2020-01-18 18:01:02	2020-01-20 15:03:58
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	17	4	1	2020-01-18 18:01:03	2020-01-20 15:03:58
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	18	5	1	2020-01-18 18:03:24	2020-01-20 15:31:41
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	19	4	1	2020-01-18 18:03:26	2020-01-20 15:35:58
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	20	7	1	2020-01-18 18:05:41	2020-01-20 15:36:45
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	21	8	1	2020-01-18 18:05:44	2020-01-20 15:36:58

/!\ On peut voir que le point de passage 00000004 par exemple possède l'ID 5 des points de passages. Cela s'explique par le fait que les points de passage commencent à 00000000. Ainsi, c'est le point de passage 00000000 qui possède l'ID 1. Points de passage et ID sont décalés de 1/!

- Retag du chef 2 sur sa base pour finir sa ronde et réponse du serveur :

```
NOUVELLE TRAME RECUE DU CLIENT
Received packet from 127.0.0.1:49562
Data: [B1:2A:B5:AA:CC:01 20000002 2020-01-20 15:37:00,Retag 2 chef fin de ronde OK,(null)]

amac=|B1:2A:B5:AA:CC:01| uid=|20000002| date=|2020-01-20| time=|15:37:00| comments=|Retag 2 chef fin de ronde OK| modeleSelect=|(null)|

PARTIE GESTION DU DEAMON

INFOS DU CLIENT
uid = 20000002
Time = 15:37:00
Date = 2020-01-20
MACadress = B1:2A:B5:AA:CC:01
Modele Selectionne= (null)

requete = SELECT `Nom` FROM `Modele_Ronde` WHERE 1

Numero du baton utilise par le client = B1:2A:B5:AA:CC:01
Etat du baton (sur le serveur) utilise par le client = 1

UN CHEF VIENT DE TAGUER SON BATON A LA BASE POUR TERMINER SA RONDE !

Information serveur : HEURE DEBUT MODELE : 15:30:0
Information serveur : HEURE FIN MODELE : 15:37:0

requete = UPDATE `Modele_Ronde` SET `Duree` = '00:07:00', `ETAT` = '1' WHERE `Nom` like 'modele2'

requete = UPDATE `Baton_Rondier` SET `ETAT` = '0' WHERE `idBaton_Rondier` like '1'

Information serveur : Le baton de rondier 1 est maintenant libre
```

Etat de la table Modele_Ronde_has_Point_Passage :

	idModele_Ronde_has_Point_Passage	Date_creation	Num	Modele_Ronde_idModele_Ronde	Point_Passage_idPoint_Passage	Date_event
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	8	2020-01-18 18:01:02	1	3	2	2020-01-20 15:02:04
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	9	2020-01-18 18:01:02	2	3	3	2020-01-20 15:03:58
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	10	2020-01-18 18:01:03	3	3	4	2020-01-20 15:03:58
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	11	2020-01-18 18:03:24	1	4	5	2020-01-20 15:31:41
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	12	2020-01-18 18:03:26	2	4	4	2020-01-20 15:35:58
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	13	2020-01-18 18:05:41	3	4	7	2020-01-20 15:36:45
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	14	2020-01-18 18:05:44	4	4	8	2020-01-20 15:36:58

Etat de la table Modele_Ronde :

	idModele_Ronde	Nom	Duree	ETAT	Ident_Agent_idIdent_Agent
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	modele1	00:05:02	1	1
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	4	modele2	00:07:00	1	2

- ✓ Le deuxième encadré concerne la création d'une instance (numéro 2) de ronde pour l'agent (garde) possédant l'uid 10000002. Cet agent démarre donc sa ronde en sélectionnant le modèle de ronde numéro 2 et tague les tableaux :

- Il tague sa base avec le bâton 3 et le serveur confirme la demande du client :

```
NOUVELLE TRAME RECUE DU CLIENT
Received packet from 127.0.0.1:38870
Data: [B1:2A:B5:AA:CC:03 10000002 2020-01-20 16:08:00,L'agent 2 démarre OK avec modele2,
2]

amac=B1:2A:B5:AA:CC:03| uid=|10000002| date=|2020-01-20| time=|16:08:00| comments=|L'ag
ent 2 démarre OK avec modele2| modeleSelect=|2|

PARTIE GESTION DU DEAMON

INFOS DU CLIENT
uid = 10000002
Time = 16:08:00
Date = 2020-01-20
MACadress = B1:2A:B5:AA:CC:03
Modele Selectionne= 2

requete = SELECT `Nom` FROM `Modele_Ronde` WHERE 1

requete = SELECT `ETAT` FROM `Baton_Rondier` WHERE `ETAT` = '1' OR `ETAT` = '2' AND `Adr
ess_Mac` = 'B1:2A:B5:AA:CC:03'

Information serveur : Il n'y a pas eu de coupure du serveur pendant une creation de mode
le ou execution d'une instance de ronde !

Numero du baton utilise par le client = B1:2A:B5:AA:CC:03
Etat du baton (sur le serveur) utilise par le client = 0

UN GARDE VIENT DE TAGUER SON BATON A LA BASE POUR COMMENCER SA RONDE !

requete = SELECT * FROM `Modele_Ronde` WHERE `Nom` = 'modele2' AND `ETAT` = '1'

requete = SELECT * FROM `Instance_Ronde` WHERE 1

Information serveur : NOMBRE D'INSTANCES EXISTANTES DANS LA BASE 1

requete = UPDATE `Baton_Rondier` SET `ETAT` = '2' WHERE `idBaton_Rondier` like '3'

Information serveur : HEURE DEBUT INSTANCE : 16:8:0

requete=INSERT INTO `Instance_Ronde` (`ETAT`,`Nom`,`Modele_Ronde_idModele_Ronde`,`Baton_
Rondier_idBaton_Rondier`,`Ident_Agent_idIdent_Agent`) VALUES('EN COURS','Instance2',(Sel
ect `idModele_Ronde` from `Modele_Ronde` WHERE `Nom` = 'modele2'),(Select `idBaton_Rondi
er` from `Baton_Rondier` WHERE `idBaton_Rondier`='3'),(Select `idIdent_Agent` from `Iden
t_Agent` WHERE `UID_RFID` = '10000002'))

Information serveur : Le baton de rondier 3 n'est plus libre
Creation d'une instance de ronde.
```

Etat de la table Instance_Ronde :

	idInstance_Ronde	Date_Creation	Etat	Nom	Duree	Modele_Ronde_idModele_Ronde	Baton_Rondier_idBaton_Rondier	Ident_Agent_idIdent_Agent
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	2020-01-18 18:37:20	TERMINE	Instance1	00:06:07	3	2	3
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	4	2020-01-18 18:58:35	EN COURS	Instance2	00:00:00	4	3	4

Etat de la table Baton_Rondier :

	idBaton_Rondier	Adress_Mac	ETAT
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	B1:2A:B5:AA:CC:01	0
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	B1:2A:B5:AA:CC:02	0
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	B1:2A:B5:AA:CC:03	2
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	4	B1:2A:B5:AA:CC:04	0
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	5	B1:2A:B5:AA:CC:05	0

- L'agent tague le point de passage 00000004 qui est bien présent en tag numéro 1 du modèle de ronde. La réponse du serveur :

```
NOUVELLE TRAME RECUE DU CLIENT
Received packet from 127.0.0.1:58758
Data: [B1:2A:B5:AA:CC:03 00000004 2020-01-20 16:09:03,Tag tableau 4 OK,(null)]

amac=|B1:2A:B5:AA:CC:03| uid=|00000004| date=|2020-01-20| time=|16:09:03| comments=|Tag
tableau 4 OK| modeleSelect=|(null)|

PARTIE GESTION DU DEAMON

INFOS DU CLIENT
uid = 00000004
Time = 16:09:03
Date = 2020-01-20
MACadress = B1:2A:B5:AA:CC:03
Modele Selectionne= (null)

requete = SELECT `Nom` FROM `Modele_Ronde` WHERE 1

Numero du baton utilise par le client = B1:2A:B5:AA:CC:03
Etat du baton (sur le serveur) utilise par le client = 2

DEMANDE DE TAG DE POINT DE PASSAGE PAR UN GARDE

requete = SELECT `Ident_Tag` FROM `Point_Passage` WHERE `idPoint_Passage` IN (SELECT `Poi
nt_Passage_idPoint_Passage` FROM `Modele_Ronde_has_Point_Passage` WHERE `Num` = '1' AND
`Modele_Ronde_idModele_Ronde`=(SELECT `idModele_Ronde` FROM `Modele_Ronde` WHERE `Nom`=
'modele2'))

Information serveur : numero du point de passage selectionne : 00000004
Information serveur : numero du point de passage attendu : 00000004

requete = SELECT `Num` FROM `Modele_Ronde_has_Point_Passage` WHERE `Modele_Ronde_idModel
e_Ronde` = (Select `idModele_Ronde` from `Modele_Ronde` where `Nom` = 'modele2')

Information serveur : Le garde en est a son point de passage numero 1 apres son tag

requete = SELECT `idBaton_Rondier` FROM `Baton_Rondier` WHERE `idBaton_Rondier` = '3'

requete=INSERT INTO `Instance_Ronde_has_Point_Passage`(`Instance_Ronde_idInstance_Ronde`
, `Point_Passage_idPoint_Passage`,`Date_event`) VALUES ((SELECT `idInstance_Ronde` FROM
`Instance_Ronde` WHERE `Nom` = 'Instance2'),(Select `idPoint_Passage` from `Point_Passag
e` where `Ident_Tag`='00000004'),'2020-01-20' '16:09:03')

Information serveur : Le point de passage 00000004 a ete tague pour l'instance de ronde

requete=INSERT INTO `Historique_Rondier`(`Baton_Rondier_idBaton_Rondier`, `Point_Passage
_idPoint_Passage`,`Date_event`) VALUES ('3',(Select `idPoint_Passage` from `Point_Passag
e` where `Ident_Tag`='00000004'),'2020-01-20' '16:09:03')

Information serveur : VOUS ETES TRACE l'historique du rondier a ete complete avec l'iden
tifiant du baton de rondier utilise pour cette instance de ronde !
```

- L'agent tague ensuite un tag le point de passage numéro 00000002 qui n'est pas présent dans le modèle de ronde (modèle 2) qu'il a sélectionné. Le modèle de ronde attend le tag du point 00000003. Un message d'erreur a donc été généré pas le serveur :

```
NOUVELLE TRAME RECUE DU CLIENT
Received packet from 127.0.0.1:38530
Data: [B1:2A:B5:AA:CC:03 00000002 2020-01-20 16:10:58,Tag tableau 2 ERREUR,(null)]

amac=|B1:2A:B5:AA:CC:03| uid=|00000002| date=|2020-01-20| time=|16:10:58| comments=|Tag
tableau 2 ERREUR| modeleSelect=|(null)|

PARTIE GESTION DU DEAMON

INFOS DU CLIENT
uid = 00000002
Time = 16:10:58
Date = 2020-01-20
MACadress = B1:2A:B5:AA:CC:03
Modele Selectionne= (null)

requete = SELECT `Nom` FROM `Modele_Ronde` WHERE 1

Numero du baton utilise par le client = B1:2A:B5:AA:CC:03
Etat du baton (sur le serveur) utilise par le client = 2

DEMANDE DE TAG DE POINT DE PASSAGE PAR UN GARDE

requete = SELECT `Ident_Tag` FROM `Point_Passage` WHERE `idPoint_Passage` IN (SELECT `Poi
nt_Passage_idPoint_Passage` FROM `Modele_Ronde_has_Point_Passage` WHERE `Num` = '2' AND
`Modele_Ronde_idModele_Ronde`=(SELECT `idModele_Ronde` FROM `Modele_Ronde` WHERE `Nom`=
'modele2'))

Information serveur : numero du point de passage selectionne : 00000002
Information serveur : numero du point de passage attendu : 00000003

requete = SELECT `Num` FROM `Modele_Ronde_has_Point_Passage` WHERE `Modele_Ronde_idModel
e_Ronde`= (Select `idModele_Ronde` from `Modele_Ronde` where `Nom` = 'modele2')

ERREUR : ATTENTION vous devez respecter les etapes du modele de ronde ! Vous DEVEZ TAGUE
R le tag : 00000003 ! Si vous avez deja tague ce tag vous avez alors termine votre ronde
!
```

Le tag de ce point de passage n'a bien sûr pas été enregistré dans la base de données.

- L'agent tague ensuite le point de passage numéro 00000003 qui est bien le deuxième dans le modèle de ronde, puis il retague sa base avant d'avoir terminé sa ronde. Le serveur génère donc une erreur :

```
NOUVELLE TRAME RECUE DU CLIENT
Received packet from 127.0.0.1:56766
Data: [B1:2A:B5:AA:CC:03 10000002 2020-01-20 16:13:07,Retag agent 2 fin instance ERREUR,
(null)]

amac=|B1:2A:B5:AA:CC:03| uid=|10000002| date=|2020-01-20| time=|16:13:07| comments=|Retag
agent 2 fin instance ERREUR| modeleSelect=|(null)|

PARTIE GESTION DU DEAMON

INFOS DU CLIENT
uid = 10000002
Time = 16:13:07
Date = 2020-01-20
MACadress = B1:2A:B5:AA:CC:03
Modele Selectionne= (null)

requete = SELECT `Nom` FROM `Modele_Ronde` WHERE 1

Numero du baton utilise par le client = B1:2A:B5:AA:CC:03
Etat du baton (sur le serveur) utilise par le client = 2

UN GARDE VIENT DE TAGUER SON BATON A LA BASE POUR TERMINER SA RONDE !

requete = SELECT `Num` FROM `Modele_Ronde_has_Point_Passage` WHERE `Modele_Ronde_idModel
e_Ronde`= (Select `idModele_Ronde` from `Modele_Ronde` where `Nom` = 'modele2')

Information serveur : Nombre de points de passages a effectuer par le garde =4
Information serveur : Etape de point de passage du modele ou en est le garde dans sa ron
de =2

ERREUR : Vous n'avez pas termine votre RONDE !
```


- L'agent tague ensuite les point de passage numéro 00000006 et 00000007 qui suivent bien le modèle de ronde. Mais celui-ci retague son dernier point de passage 00000007. Une erreur est alors générée par le serveur :

```
NOUVELLE TRAME RECUE DU CLIENT
Received packet from 127.0.0.1:41102
Data: [B1:2A:B5:AA:CC:03 00000007 2020-01-20 16:14:45,Tag tableau 7 ERREUR,(null)]

amac=|B1:2A:B5:AA:CC:03| uid=|00000007| date=|2020-01-20| time=|16:14:45| comments=|Tag
tableau 7 ERREUR| modeleSelect=|(null)|

PARTIE GESTION DU DEAMON

INFOS DU CLIENT
uid = 00000007
Time = 16:14:45
Date = 2020-01-20
MACadress = B1:2A:B5:AA:CC:03
Modele Selectionne= (null)

requete = SELECT `Nom` FROM `Modele_Ronde` WHERE 1

Numero du baton utilise par le client = B1:2A:B5:AA:CC:03
Etat du baton (sur le serveur) utilise par le client = 2

DEMANDE DE TAG DE POINT DE PASSAGE PAR UN GARDE

requete = SELECT `Ident_Tag` FROM `Point_Passage` WHERE `idPoint_Passage` IN (SELECT `Poi
nt_Passage_idPoint_Passage` FROM `Modele_Ronde_has_Point_Passage` WHERE `Num` = '5' AND
`Modele_Ronde_idModele_Ronde`=(SELECT `idModele_Ronde` FROM `Modele_Ronde` WHERE `Nom` =
'modele2'))

Information serveur : numero du point de passage selectionne : 00000007
Information serveur : numero du point de passage attendu : 00000007

requete = SELECT `Num` FROM `Modele_Ronde_has_Point_Passage` WHERE `Modele_Ronde_idModel
e_Ronde`=(Select `idModele_Ronde` from `Modele_Ronde` where `Nom` = 'modele2')

ERREUR : ATTENTION vous devez respecter les etapes du modele de ronde ! Vous DEVEZ TAGUE
R le tag : 00000007 ! Si vous avez deja tague ce tag vous avez alors termine votre ronde
!
```

Etat de la table Historique_Rondier :

	IdPoint_Passage_has_Baton_Rondier	Point_Passage_IdPoint_Passage	Baton_Rondier_IdBaton_Rondier	Date_Creation	Date_event
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	15	2	1	2020-01-18 18:01:02	2020-01-20 15:02:04
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	16	3	1	2020-01-18 18:01:02	2020-01-20 15:03:58
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	17	4	1	2020-01-18 18:01:03	2020-01-20 15:03:58
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	18	5	1	2020-01-18 18:03:24	2020-01-20 15:31:41
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	19	4	1	2020-01-18 18:03:26	2020-01-20 15:35:50
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	20	7	1	2020-01-18 18:05:41	2020-01-20 15:36:45
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	21	8	1	2020-01-18 18:05:44	2020-01-20 15:36:50
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	22	2	2	2020-01-18 18:37:17	2020-01-20 16:02:03
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	23	3	2	2020-01-18 18:37:17	2020-01-20 16:03:58
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	24	4	2	2020-01-18 18:37:18	2020-01-20 16:05:58
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	25	5	3	2020-01-18 18:42:03	2020-01-20 16:09:03
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	26	4	3	2020-01-18 18:43:16	2020-01-20 16:12:50
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	27	7	3	2020-01-18 18:45:32	2020-01-20 16:14:02
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	28	8	3	2020-01-18 18:45:39	2020-01-20 16:14:45

- L'agent 2 retague sa base avec son bâton de rondier pour terminer son instance de ronde. Réponse du serveur :

```

NOUVELLE TRAME RECUE DU CLIENT
Received packet from 127.0.0.1:45573
Data: [B1:2A:B5:AA:CC:03 10000002 2020-01-20 16:15:07,Retag agent 2 fin instance OK,(null)]

amac=|B1:2A:B5:AA:CC:03| uid=|10000002| date=|2020-01-20| time=|16:15:07| comments=|Retag agent 2 fin instance OK| modeleSelect=|(null)|

PARTIE GESTION DU DEAMON

INFOS DU CLIENT
uid = 10000002
Time = 16:15:07
Date = 2020-01-20
MACadress = B1:2A:B5:AA:CC:03
Modele Selectionne= (null)

requete = SELECT `Nom` FROM `Modele_Ronde` WHERE 1

Numero du baton utilise par le client = B1:2A:B5:AA:CC:03
Etat du baton (sur le serveur) utilise par le client = 2

UN GARDE VIENT DE TAGUER SON BATON A LA BASE POUR TERMINER SA RONDE !

requete = SELECT `Num` FROM `Modele_Ronde_has_Point_Passage` WHERE `Modele_Ronde_idModele_Ronde` = (Select `idModele_Ronde` from `Modele_Ronde` where `Nom` = 'modele2')

Information serveur : Nombre de points de passages a effectuer par le garde =4
Information serveur : Etape de point de passage du modele ou en est le garde dans sa ronde =4

Information serveur : HEURE DEBUT INSTANCE : 16:8:0
Information serveur : HEURE FIN INSTANCE : 16:15:7

requete = UPDATE `Instance_Ronde` SET `Duree` = '00:07:07', `Etat` = 'TERMINE' WHERE `Nom` like 'Instance2'

requete = UPDATE `Baton_Rondier` SET `ETAT` = '0' WHERE `idBaton_Rondier` like '3'

Information serveur : Le baton de rondier 3 est maintenant libre !
  
```

Etat de la table Instance_Ronde_has_Point_Passage :

	idInstance_Ronde_has_Point_Passage	Point_Passage_idPoint_Passage	Instance_Ronde_idInstance_Ronde	Date_creation	Date_event
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	8	2	3	2020-01-18 18:37:17	2020-01-20 16:02:03
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	9	3	3	2020-01-18 18:37:17	2020-01-20 16:03:58
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	10	4	3	2020-01-18 18:37:18	2020-01-20 16:05:58
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	11	5	4	2020-01-18 18:42:03	2020-01-20 16:09:03
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	12	4	4	2020-01-18 18:43:16	2020-01-20 16:12:58
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	13	7	4	2020-01-18 18:45:32	2020-01-20 16:14:02
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	14	8	4	2020-01-18 18:45:39	2020-01-20 16:14:45

Etat de la table Instance_Ronde :

	idInstance_Ronde	Date_Creation	Etat	Nom	Duree	Modele_Ronde_idModele_Ronde	Baton_Rondier_idBaton_Rondier	Ident_Agent_idIdent_Agent
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	2020-01-18 18:37:20	TERMINE	Instance1	00:06:07	3	2	3
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	4	2020-01-18 18:47:31	TERMINE	Instance2	00:07:07	4	3	4

L'agent a donc terminé sa ronde nommée Instance2, en 7min07 !

Conclusion

Nous avons donc pu par ce projet en apprendre plus en programmation C, sur la mise en place d'un client/serveur en connexion udp. Ce projet aura été également très enrichissant pour l'apprentissage du langage SQL et de la mise place et de la gestion d'une base de données sur phpMyAdmin.

Nous sommes assez satisfaits du rendu de ce projet que nous pensons avoir mené à bout après un très long travail. Le seul point qui restait à traiter mais qui n'était pas demandé, était la gestion et l'affichage des rondes sur une page web. Beaucoup de temps a été passé sur le serveur udp pour mettre en place le système de gestion des rondes et de gestion des erreurs. Il a fallu beaucoup réfléchir en amont sur les types de requêtes à effectuer et en déduire un algorithme adapter pour gérer les modèles, instances et les erreurs qui peuvent être rencontrés par un garde ou un chef lorsqu'il tag son bâton de rondier.

Un projet intéressant et très prenant qui nous a fait monter en compétences.