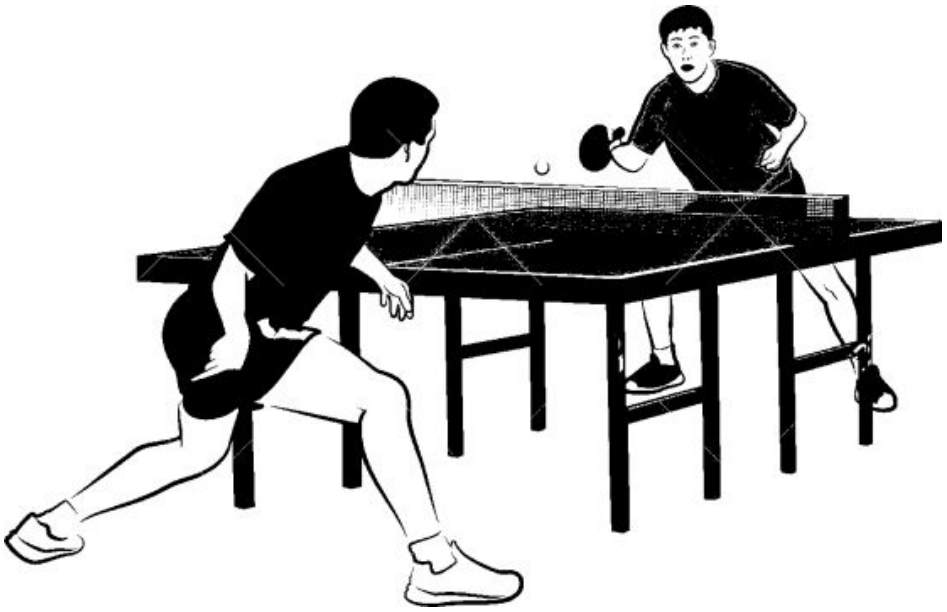
Table Tennis Performance Analyser

Projet BTS SN Lasalle - 2018

Terminal Mobile



Revue finale

25/05/2018

v1.0

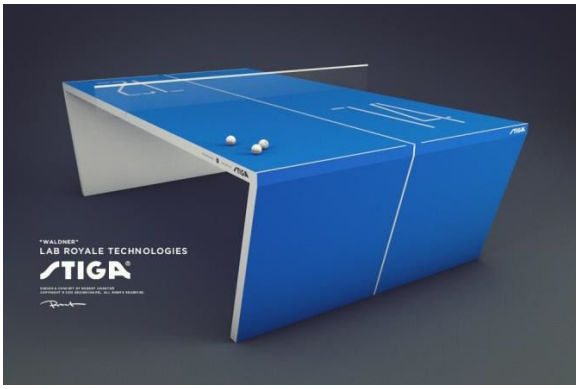
SMANIOTTO Nathan

SOMMAIRE

1 - Présentation générale du système supportant le projet

Le tennis de table, comme tout sport de raquette, est un sport complet tant sur le plan physique que mental. Il requiert un entraînement rigoureux et régulier afin de pouvoir se perfectionner. Les joueurs comme les entraîneurs ont besoin de connaître les performances en jeux et lors des phases d'entraînement. De nombreux entraîneurs ont constaté que le taux de réussite et la précision du coup sont les clefs de la réussite. Un système, qui mettrait en évidence les résultats d'un joueur, lui permettra d'optimiser au mieux ses performances.

2 - Analyse de l'existant



La « Waldner » est une table de ping pong de nouvelle génération qui a été conçue pour intégrer un système informatique de pointe.
La table dispose d'une surface sensible au contact humain et à celui des balles de ping-pong. La table est également d'un noyau informatique de type Mac Pro 12 qui observe le jeu.
Sous la surface, on trouve deux Quad-Core Intel Xeon cadencés à 2.4GHz, une carte graphique ATI Radeon HD 5870, une connexion Wi-Fi et 12 haut-parleurs/micro Bose. Les dimensions de la table sont de 2,74 m de long, 1,52 m de large, 76 cm de haut et 5 cm d'épaisseur. La surface, qui est écran écran, est faite d'acrylique très lisse et à faible coefficient de frottement. L'écran a une résolution de 2880 x 1800 pixels.
Après et pendant le match, de nombreux éléments statistiques peuvent être obtenus sur la table. Les joueurs peuvent par exemple comparer les zones de toucher de balles (où chaque impact est matérialisé par un cercle et faisant mention d'informations comme la vitesse et la trajectoire. Toutes les données sont stockées et peuvent être visualisées en en temps réel sur des périphériques comme l'iPad et l'iPhone.

3 - Expression du besoin

Malgré la fiabilité de la table « Waldner » , elle ne convient pas à tous les publics. En effet, sa précision a un coût et il est conséquent. De plus, la visualisation des données nécessite un périphérique de type iPad ou iPhone donc non compatible sous une plateforme Android (mobile, tablette, télévision). C'est pour cela qu'un système moins sophistiqué et donc moins coûteux est nécessaire au sein de ce marché.

Ce besoin s'appuie d'autant plus sur la démarche que le *Ping Pong Club Sorgues* a effectuée avec l'établissement Lasalle Avignon, dont un de leur membre y est professeur. Ils ont donc pu communiquer leur idée de projet, à l'origine de ce système :

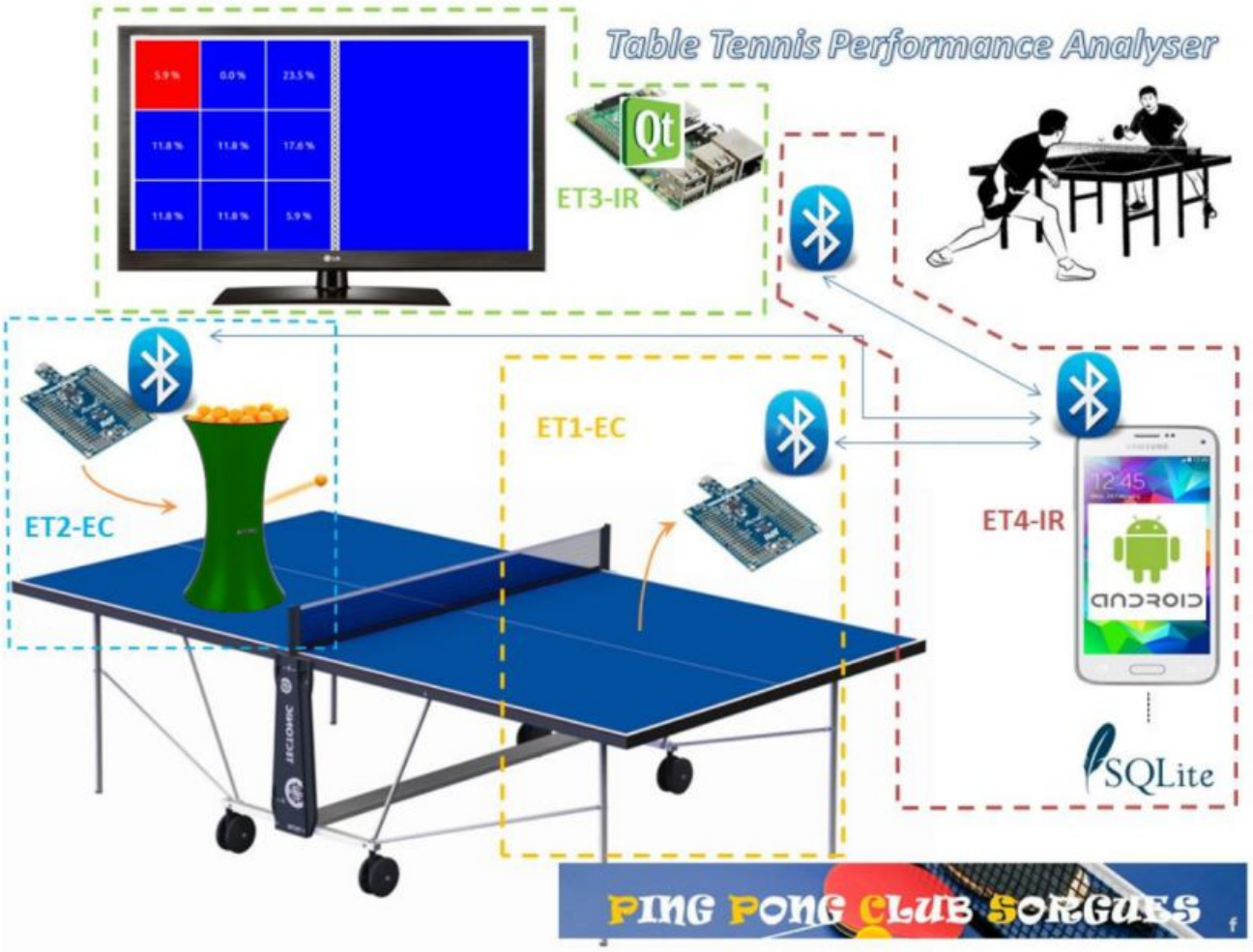
Ce système doit permettre une analyse des performances du joueur (côté relanceur). Il doit proposer une phase d'entraînement adaptée au niveau du joueur, puis de détecter l'impact des balles afin d'afficher le rythme de jeu, la précision, le pourcentage de réussite. La zone d'impact (côté distributeur) est identifiée sur un écran de télévision en fin d'exercice. Le pourcentage de balles dans chacune des zones, le rythme de jeu et le pourcentage de réussite sont disponibles en fin d'exercice. Le joueur lance un exercice spécifique et pourra connaître son évolution individuelle.

4 - Présentation globale du projet

4.1 - Points généraux du système

Le système ayant pour but d'analyser les performances individuelles d'un joueur, il se doit donc de pouvoir lancer une phase d'entraînement spécifique aux besoins de celui-ci. Une séance sera donc paramétrable (nombre de balles, effet, vitesse des balles, etc…) et tout au long de celle-ci, les impacts des balles ainsi que les zones touchées seront déterminées. Dans le but d'une possible amélioration des capacités du joueur, un retour visuel devra être possible afin de pouvoir observer son évolution personnelle.

4.2 - Synoptique du système



Ce schéma fourni par le cahier des charges nous indique clairement les différents modules qui feront partie de ce système, ainsi que leur mode de communication (essentiellement **Bluetooth**) et leurs relations :

- Module ET1-EC : Ce module qui a pour fonctionnalité de détecter l'impact d'une balle sur la table sera composé de capteurs, permettant de déterminer et communiquer, en Bluetooth via une carte Arduino, la zone impactée.

- Module ET2-EC : Ce module est composé d'un robot iPong permettant de lancer des balles selon des caractéristiques données (effet, puissance, etc…).
  La communication des caractéristiques se fait en Bluetooth grâce à une carte Arduino.

- Module ET3-IR : Ce module permet la visualisation des informations de la séance. En effet, à l'aide d'un écran de télévision, les impacts reçus via Bluetooth grâce à une carte Arduino peuvent être visualisés en temps réel.
  De plus, les impacts sont traités et les statistiques de la séance en cours sont calculés et affichés.

- Module ET4-IR : Ce module composé d'une application sous Android, est le point névralgique du système.
  En effet toutes les informations passeront par celui-ci. Il permet notamment la paramétrage d'une séance ainsi que le démarrage de celle-ci, puis la visualisation des séances précédemment jouées.
  L'application communiquera en Bluetooth avec tous les autres modules du système.

4.3 - Fonctions issues du cahier des charges

Premièrement, grâce au robot lanceur de balles ainsi qu'au terminal mobile, il sera possible de régler les paramètres d'une séance qui sont les suivants :

- Zone du robot lanceur de balles,
- Zone d'objectif du joueur permettant le calcul du pourcentage de réussite (si aucune zone n'est sélectionnée, la table entière devient l'objectif),
- Fréquence d'envoi des balles,
- Effet des balles envoyées (coupé, lifté, sans effet),
- Vitesse des balles,
- Rotation du robot lanceur de balles.

Ensuite, les capteurs placés sur la table permettront de pouvoir localiser les impacts des balles.
Enfin, ces impacts seront retransmis à l'écran et au terminal mobile afin de pouvoir visualiser en temps réel les statistiques du joueur.

4.4 - Répartition des modules

La répartition des modules s'est faite telle quelle :

Etudiant	Spécialité	Module
VIDAL Damien	EC	Robot Lanceur
GRENOD Pierre	EC	Capteurs Table
RACAMOND Adrien	IR	Ecran TV
SMANIOTTO Nathan	IR	Terminal Mobile

4.5 - Organisation commune au sein du projet

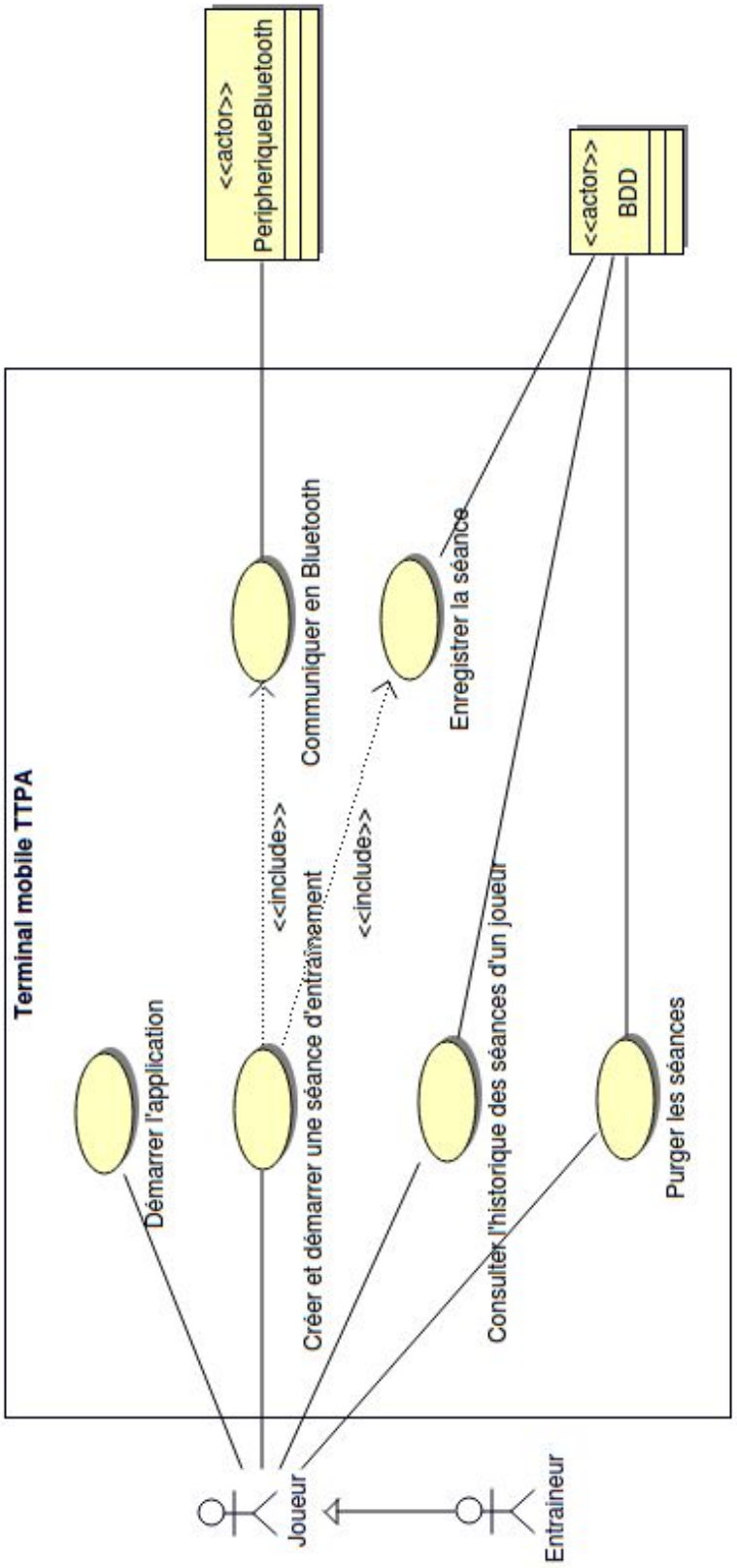
Afin de permettre une meilleur organisation, communication et gestion des fichiers et codes sources, nous avons utilisé **Subversion** qui est un logiciel libre de gestion de versions. Ainsi qu'un espace de stockage commun (**NAS**) pour tous les documents ressources. Enfin, pour la communication matériel entre les différents membres du projet, nous avons mis au point un **protocole de communication**, dont je détaillerai les caractéristiques dans le développement de la partie Terminal Mobile.

5 - Implication personnelle

5.1 - Réalisation personnelle au sein du projet

Les fonctionnalités principales du module Terminal Mobile sont les suivantes :

- Régler les paramètres d'une séance,
- Communiquer les informations aux autres modules en Bluetooth grâce à des trames,
- Traiter les résultats d'une séance afin d'en générer le taux de réussite,
- Enregistrer une séance afin de pouvoir consulter ses caractéristiques ultérieurement.


Terminal mobile TTPA
<<actor>>
PeripheriqueBluetooth
<<actor>>
BDD
Communiquer en Bluetooth
Enregistrer la séance
<<include>>
<<include>>
Démarrer l'application
Créer et démarrer une séance d'entraînement
Consulter l'historique des séances d'un joueur
Purger les séances
Joueur
Entraineur

Le diagramme de cas d'utilisation du Terminal Mobile présent ci-dessus présente la vision du joueur ou d'un entraîneur par rapport au système. En effet, de son point de vue, il aura la possibilité de *Démarrer l'application*, *Créer et démarrer une séance d'entraînement* en l'ayant préalablement paramétrée ou non (paramètres par défaut), ensuite, après avoir joué sa première séance, il pourra *Consulter l'historique des séances* et enfin *Purger l'historique des séances*.
Cependant, même si le joueur n'en a pas conscience au premier abord, ces fonctions nécessitent de pouvoir se connecter aux appareils Bluetooth du système, ainsi que de pouvoir enregistrer les données de la séance. Dans le cas échéant, le joueur ne pourra ni démarrer une séance, ni consulter son historique de séances.

5.2 - Planification personnelle et liste des tâches

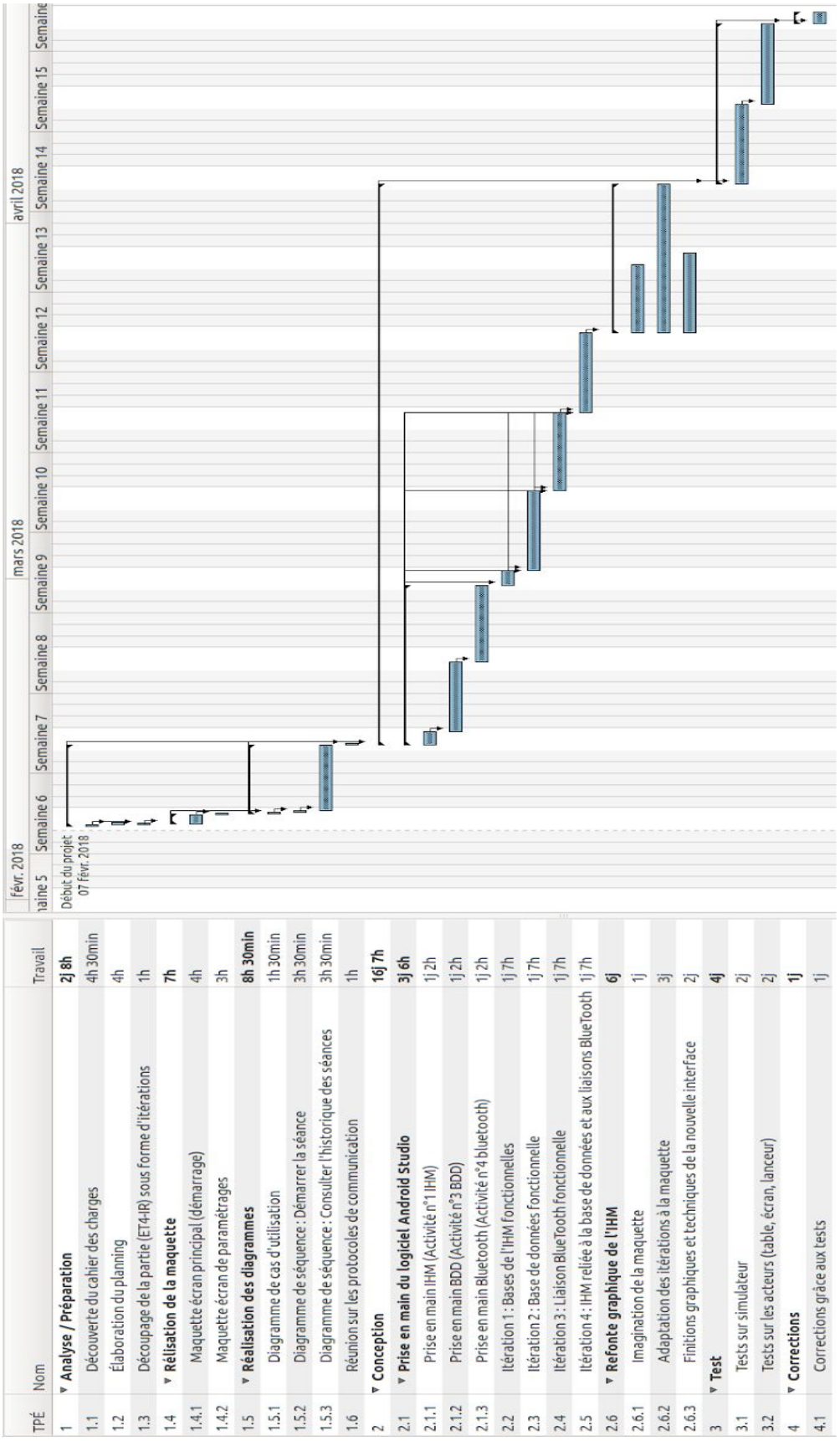
La planification (du 07/02/18 au 19/04/18) ci-dessous a été structurée sous différentes phases :

- La première phase consiste à analyser les caractéristiques fournies par le cahier des charges, mais aussi de préparer l'organisation du projet, notamment la planification.

- La deuxième grande phase consiste à concevoir l'application mobile, cependant, cela nécessite une prise en main des technologies utilisées.

- Une période d'essais du programme a été prévue, même si chaque modification est testée après changement du programme afin de valider son bon fonctionnement.

- De même pour les correctifs, la majeur partie des correctifs s'est effectuée lors de la conception.



TPÉ	Nom	Travail
1	**Analyse / Préparation**	**2j 8h**
1.1	Découverte du cahier des charges	4h 30min
1.2	Élaboration du planning	4h
1.3	Découpage de la partie (ET4-IR) sous forme d'itérations	1h
1.4	**Réllisation de la maquette**	**7h**
1.4.1	Maquette écran principal (démarrage)	4h
1.4.2	Maquette écran de paramétrages	3h
1.5	**Réalisation des diagrammes**	**8h 30min**
1.5.1	Diagramme de cas d'utilisation	1h 30min
1.5.2	Diagramme de séquence : Démarrer la séance	3h 30min
1.5.3	Diagramme de séquence : Consulter l'historique des séances	3h 30min
1.6	Réunion sur les protocoles de communication	1h
2	**Conception**	**16j 7h**
2.1	**Prise en main du logiciel Android Studio**	**3j 6h**
2.1.1	Prise en main IHM (Activité n°1 IHM)	1j 2h
2.1.2	Prise en main BDD (Activité n°3 BDD)	1j 2h
2.1.3	Prise en main Bluetooth (Activité n°4 bluetooth)	1j 2h
2.2	Itération 1 : Bases de l'IHM fonctionnelles	1j 7h
2.3	Itération 2 : Base de données fonctionnelle	1j 7h
2.4	Itération 3 : Liaison BlueTooth fonctionnelle	1j 7h
2.5	Itération 4 : IHM reliée à la base de données et aux liaisons BlueTooth	1j 7h
2.6	**Refonte graphique de l'IHM**	**6j**
2.6.1	Imagination de la maquette	1j
2.6.2	Adaptation des itérations à la maquette	3j
2.6.3	Finitions graphiques et techniques de la nouvelle interface	2j
3	**Test**	**4j**
3.1	Tests sur simulateur	2j
3.2	Tests sur les acteurs (table, écran, lanceur)	2j
4	**Corrections**	**1j**
4.1	Corrections grâce aux tests	1j

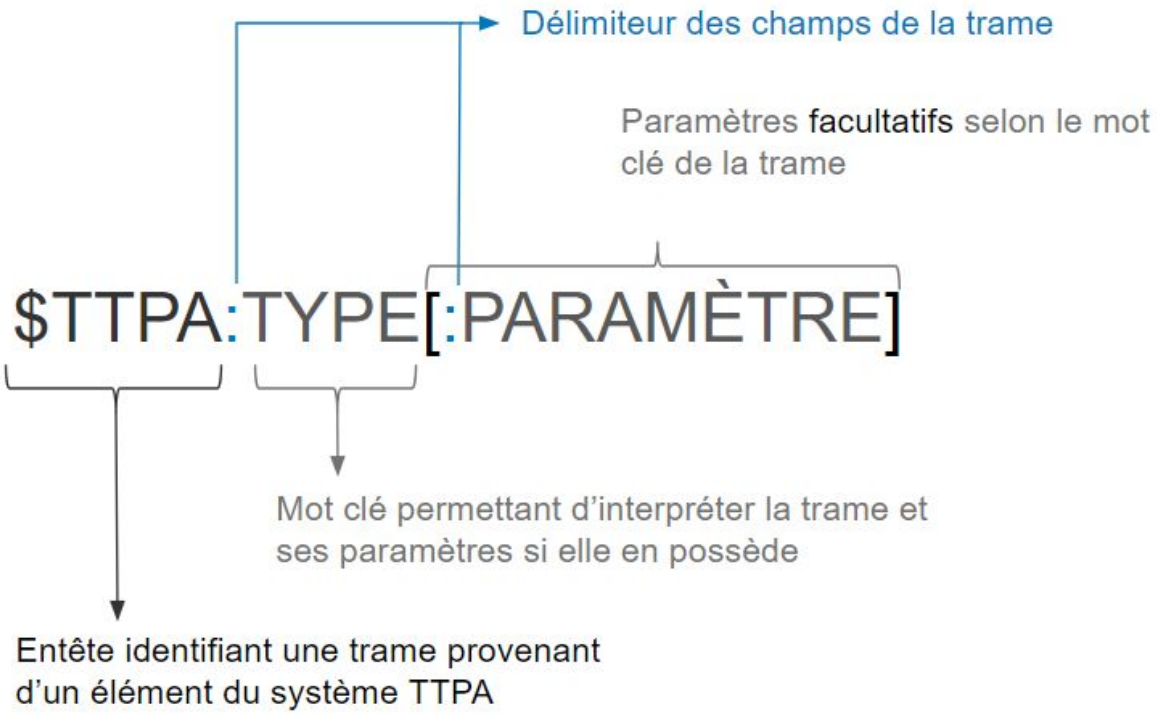
5.3 - Ressources logicielles

Fonctionnalité	Elément utilisé	Version
Système d'exploitation du poste de développement	GNU/LINUX	3.8.0-34-generic #49~precise1-Ubuntu
Logiciel de planification	Planner	0.14.5
Génération des diagrammes	BOUML	7.4
Gestion de versions	Subversion	1.6.17
Environnement de développement	Android Studio	2.3
Génération de la documentation	Doxygen	1.7.6.1
Tablette utilisée pour les essais et démonstrations	Tablette Samsung	SM-T530

Caractéristiques techniques de la tablette **SM-T530** utilisée pour les essais et démonstrations :

Composant	Description
Système d'exploitation	Android 5.0.2
Processeur	Snapdragon 400 Quad-core 1.2GHz
Mémoire RAM	1.5Go
Carte graphique	Adreno 305

5.4 - Protocole de communication Bluetooth utilisé



Toute l'équipe du projet s'est mis d'accord sur un protocole de communication utilisant ces caractéristiques :

- **$TTPA** : entête permettant d'identifier une trame en provenance d'un appareil du système TTPA.
- **:** : délimiteur permettant de séparer les champs.
- **TYPE** : étant une chaîne de caractère, soit un mot clé permettant d'identifier le type de la trame.
- Des paramètres supplémentaires entourés du délimiteur **:** peuvent être ajoutés, selon le type de la trame et les informations que l'on veut communiquer.
- **\r\n** : étant un retour chariot permettant de signaler la fin de la trame.

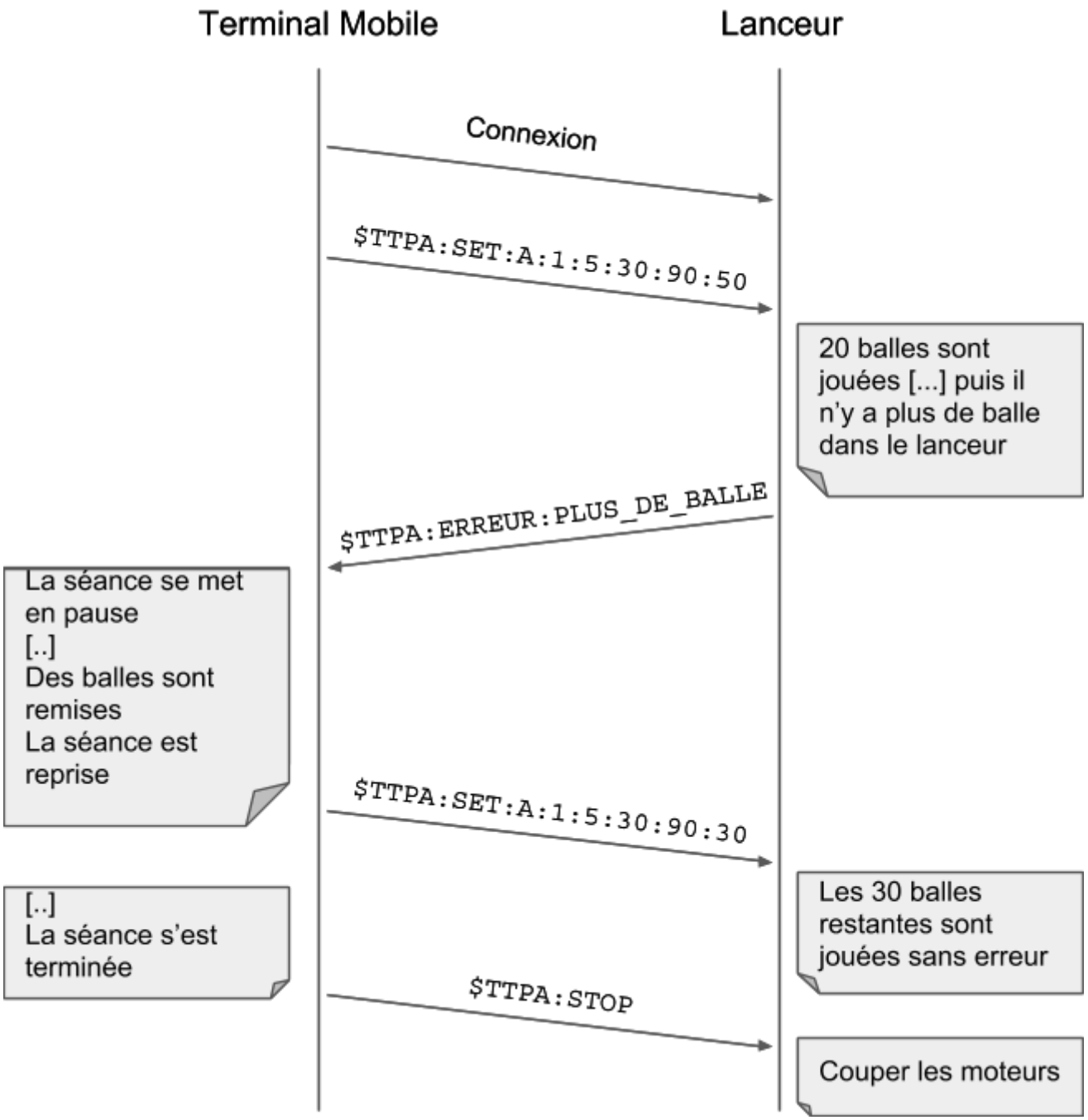
5.5 - Trames reçues et envoyées par le Terminal Mobile

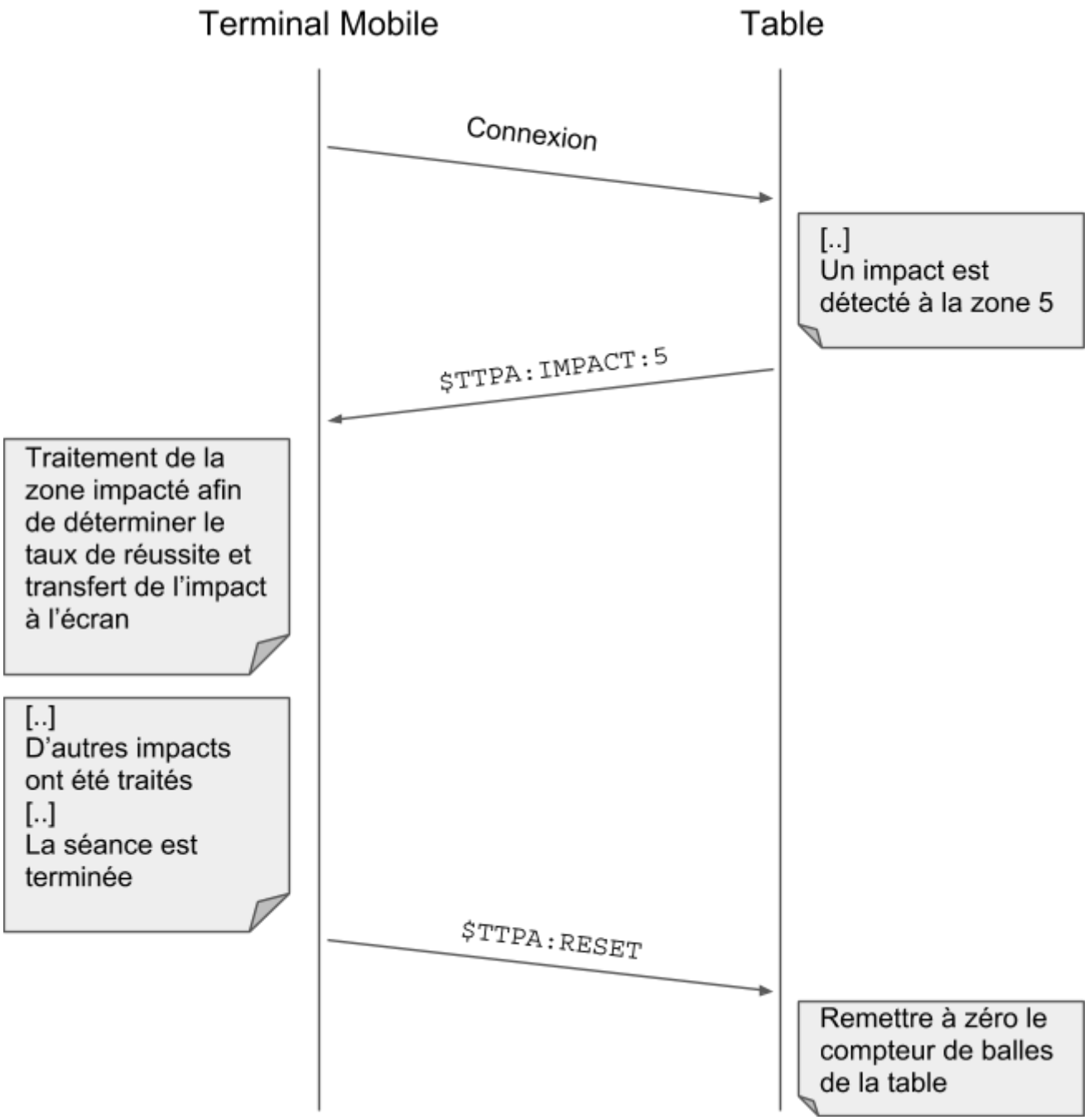
5.5.1 - Liste des trames

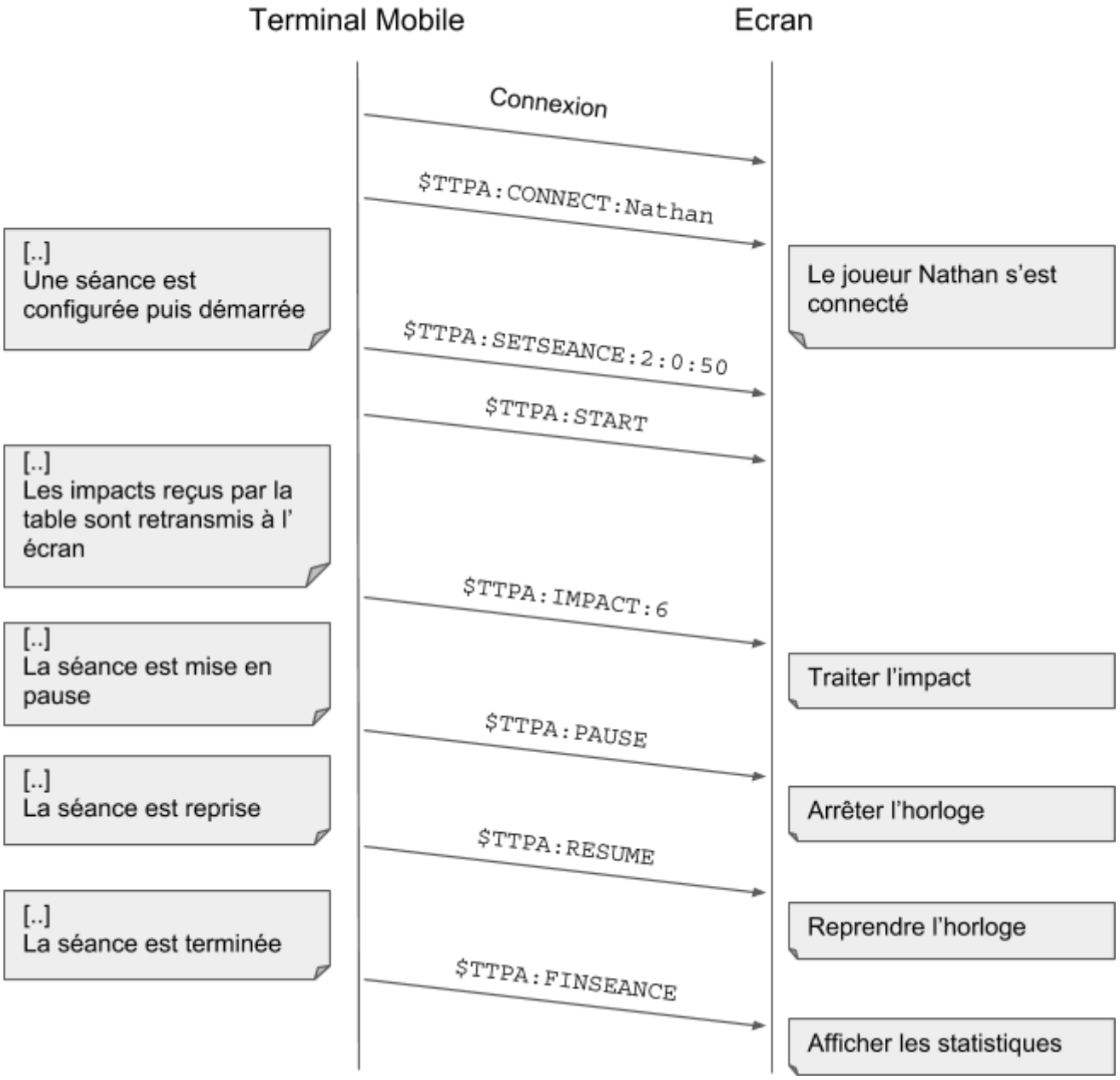
Source	Destination	Trame	Fonctionnalité
Table	Terminal Mobile	`$TTPA:IMPACT:z`	Identification de la zone touchée par la balle par des numérotations allant de **0** à **9** pour z.
Table	Terminal Mobile	`$TTPA:FAUTE`	Signale que le joueur n'a pas touché de zone, donc qu'il a fait une faute.
Lanceur	Terminal Mobile	`$TTPA:ERREUR:Description`	Signaler une erreure liée au robot, l'erreur peut être un bourrage, un manque de balle, voire une erreure liée au paramétrage
Terminal Mobile	Lanceur	`$TTPA:SET:Effet:IntensiteEffet:Puissance:Frequence:Rotation:NbBalles` Avec : Effet: **S**ans effet, **C**oupé, **L**ifté IntensiteEffet: de **1** à **9** Puissance: de **1** à **9** Fréquence: de **30** à **60** balles/min Rotation: de **0** à **180**° NbBalles: de **5** à **100** balles	Configure les paramètres d'une séance
Terminal Mobile	Lanceur	`$TTPA:STOP`	Arrêter de lancer des balles (couper les moteurs)

Source	Destination	Trame	Fonctionnalité
Terminal Mobile	Table	`$TTPA:RESET`	Remettre à zéro le compteur de balles
Terminal Mobile	Ecran	`$TTPA:CONNECT:Nom`	Affiche le champ Nom ,correspondant au nom du joueur, à l'écran
Terminal Mobile	Ecran	`$TTPA:RESET`	Remet à zéro les informations de la table et de la séance en cours
Terminal Mobile	Ecran	`$TTPA:PAUSE`	Met la séance en cours en pause
Terminal Mobile	Ecran	`$TTPA:RESUME`	Reprend la séance en cours
Terminal Mobile	Ecran	`$TTPA:FINSEANCE`	Signale la fin d'une séance
Terminal Mobile	Ecran	`$TTPA:START`	Signale le début d'une séance
Terminal Mobile	Ecran	`$TTPA:IMPACT:z`	Identification de la zone touchée par la balle par des numérotations allant de 0 à 9 pour z.
Terminal Mobile	Ecran	`$TTPA:SETSEANCE:Pos_robot:Pos_objectif:NbBalles:` Avec : Pos_robot: de 0 à 9 Pos_objectif: de 0 à 9 NbBalles: de 5 à 100 balles	Réglage des paramètres de la séance

5.5.2 - Aperçus des échanges entre les appareils




Terminal Mobile
Table
Connexion
[..]
Un impact est détecté à la zone 5
$TTPA:IMPACT:5
Traitement de la zone impacté afin de déterminer le taux de réussite et transfert de l'impact à l'écran
[..]
D'autres impacts ont été traités
[..]
La séance est terminée
$TTPA:RESET
Remettre à zéro le compteur de balles de la table

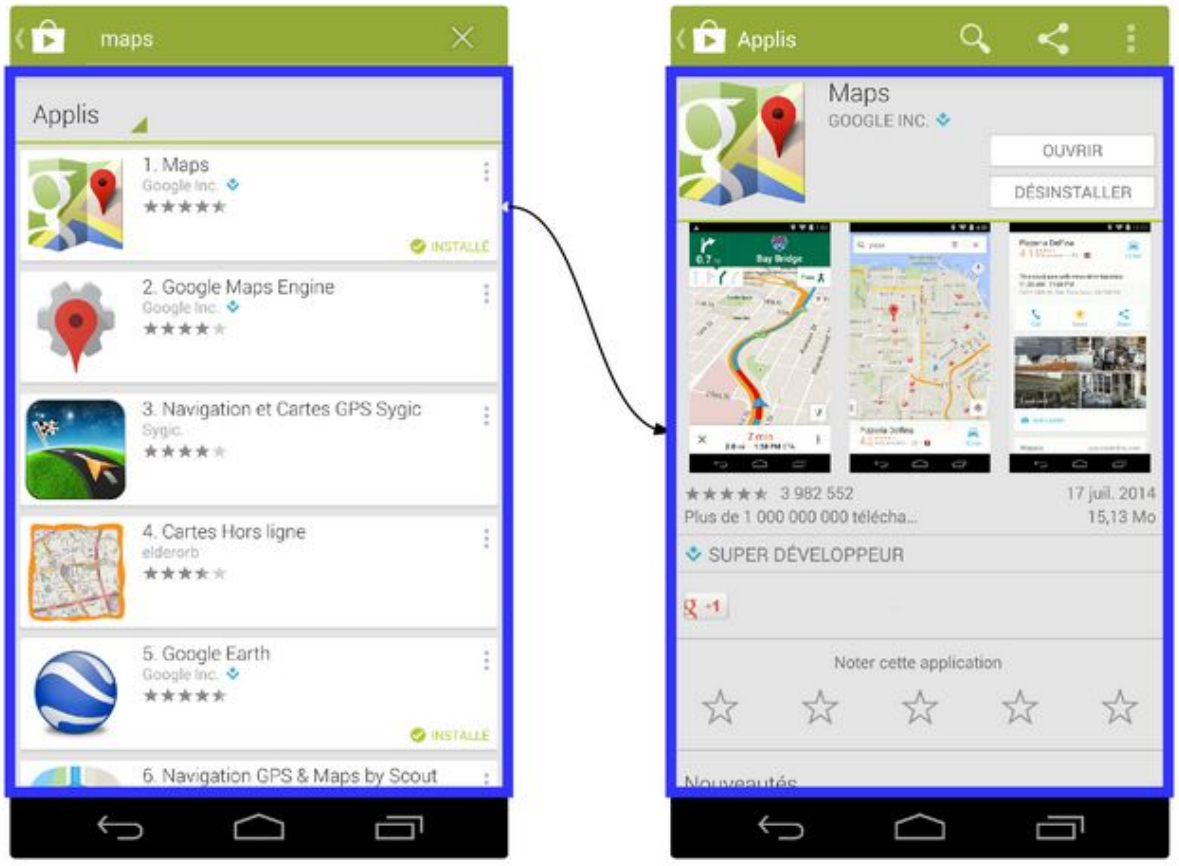

Terminal Mobile
Ecran
Connexion
$TTPA:CONNECT:Nathan
[..]
Une séance est configurée puis démarrée
Le joueur Nathan s'est connecté
$TTPA:SETSEANCE:2:0:50
$TTPA:START
[..]
Les impacts reçus par la table sont retransmis à l' écran
$TTPA:IMPACT:6
[..]
La séance est mise en pause
Traiter l'impact
$TTPA:PAUSE
[..]
La séance est reprise
Arrêter l'horloge
$TTPA:RESUME
[..]
La séance est terminée
Reprendre l'horloge
$TTPA:FINSEANCE
Afficher les statistiques

5.6 - Architecture Android

Une application Android est composée de plusieurs fenêtres.
En effet, si on effectue une recherche, une liste de résultats s'affichera dans une première fenêtre et si on clique sur un résultat, une nouvelle fenêtre s'ouvre pour nous afficher la page de présentation de l'application sélectionnée.
Au final, on peut remarquer qu'une application est un assemblage de fenêtres entre lesquelles il est possible de naviguer.
Ces différentes fenêtres sont appelées des “activités”.

5.6.1 - Explication du terme “Activité”

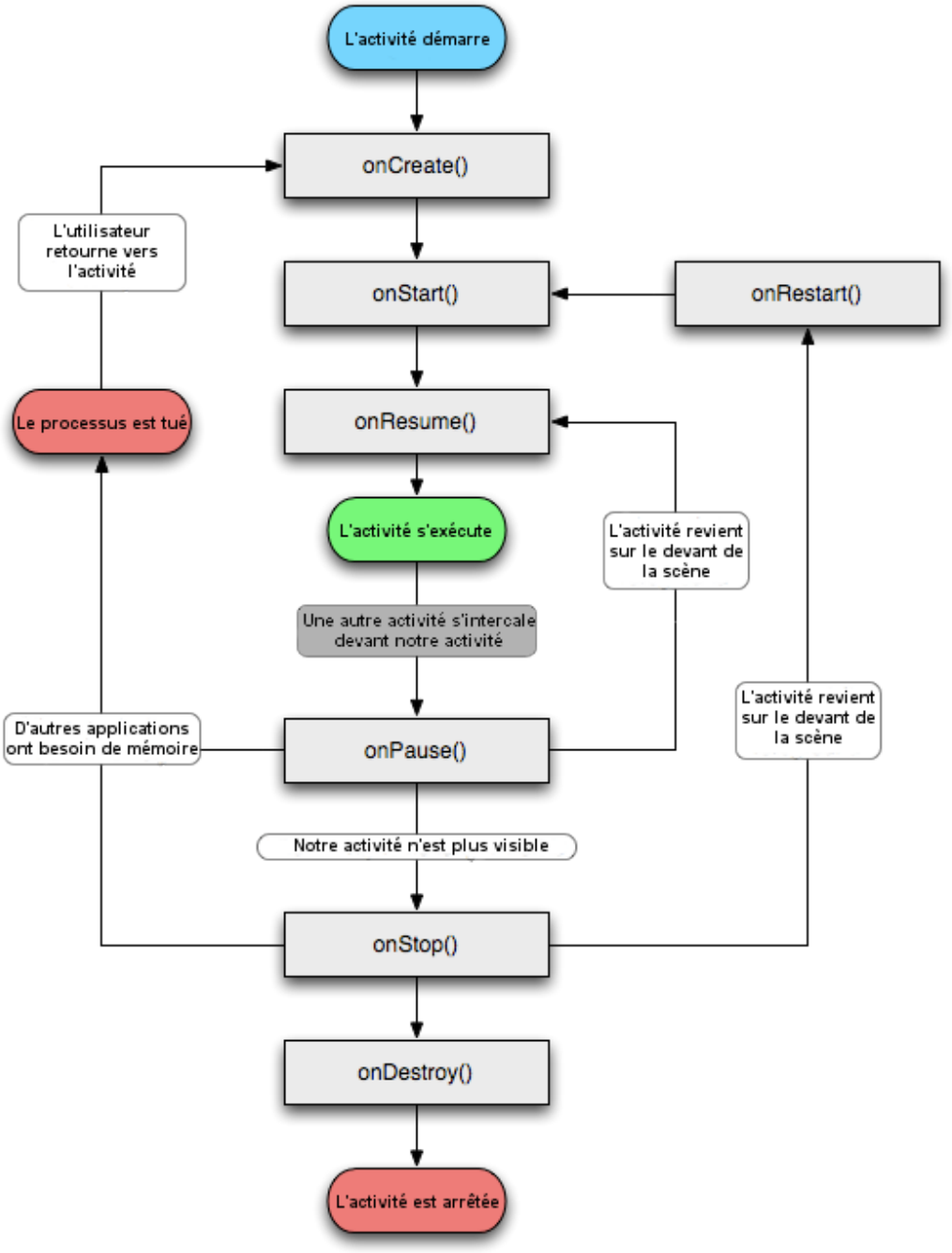
Un moyen efficace de différencier des activités est de comparer leur interface graphique : si elles sont radicalement différentes, c'est qu'il s'agit d'activités différentes. De plus, comme une activité remplit tout l'écran, une application ne peut en afficher qu'une à la fois. La figure suivante illustre ce concept.



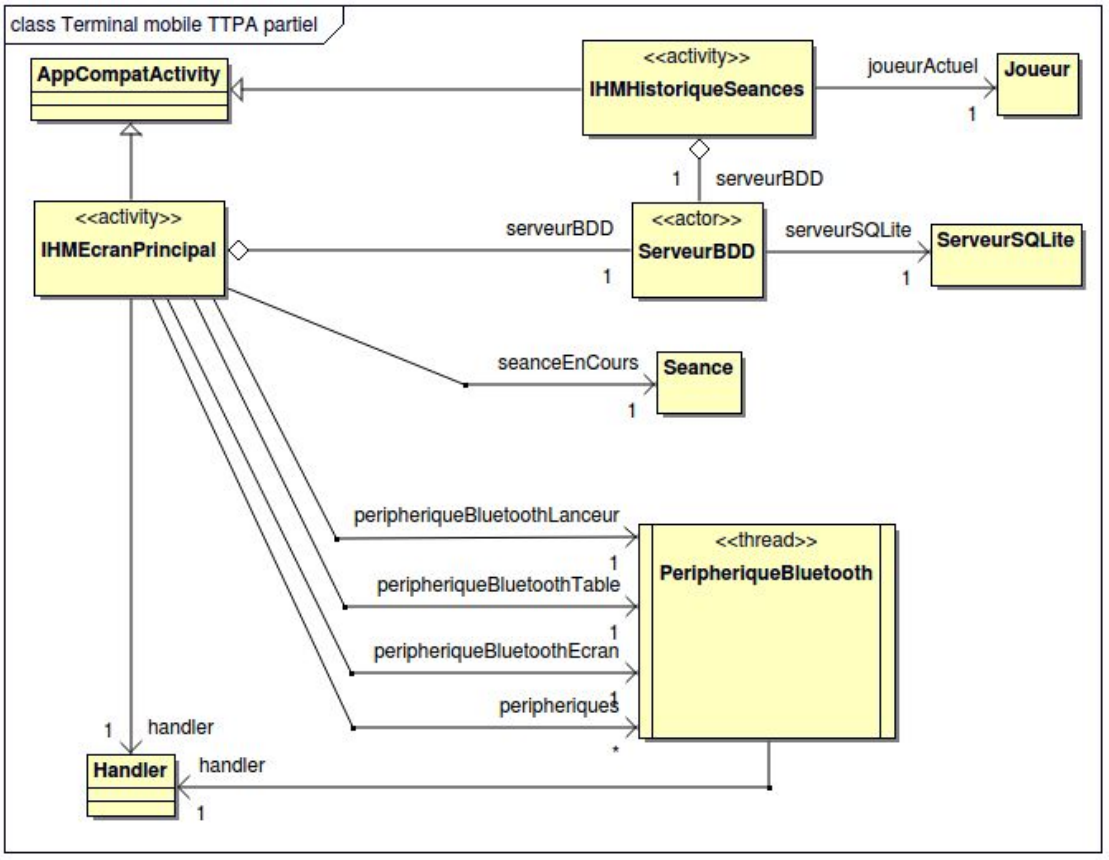
En cliquant sur un élément de la liste à gauche, on ouvre une nouvelle activité.

5.6.2 - Cycle de vie d'une activité

Une activité n'a pas de contrôle direct sur son propre état, il s'agit plutôt d'un cycle rythmé par les interactions avec le système et d'autres applications.
Voici un schéma qui présente ce que l'on appelle **le cycle de vie d'une activité**, c'est-à-dire qu'il indique les étapes que va traverser une activité pendant sa vie, de sa naissance à sa mort.
Chaque étape du cycle est représentée par une méthode.



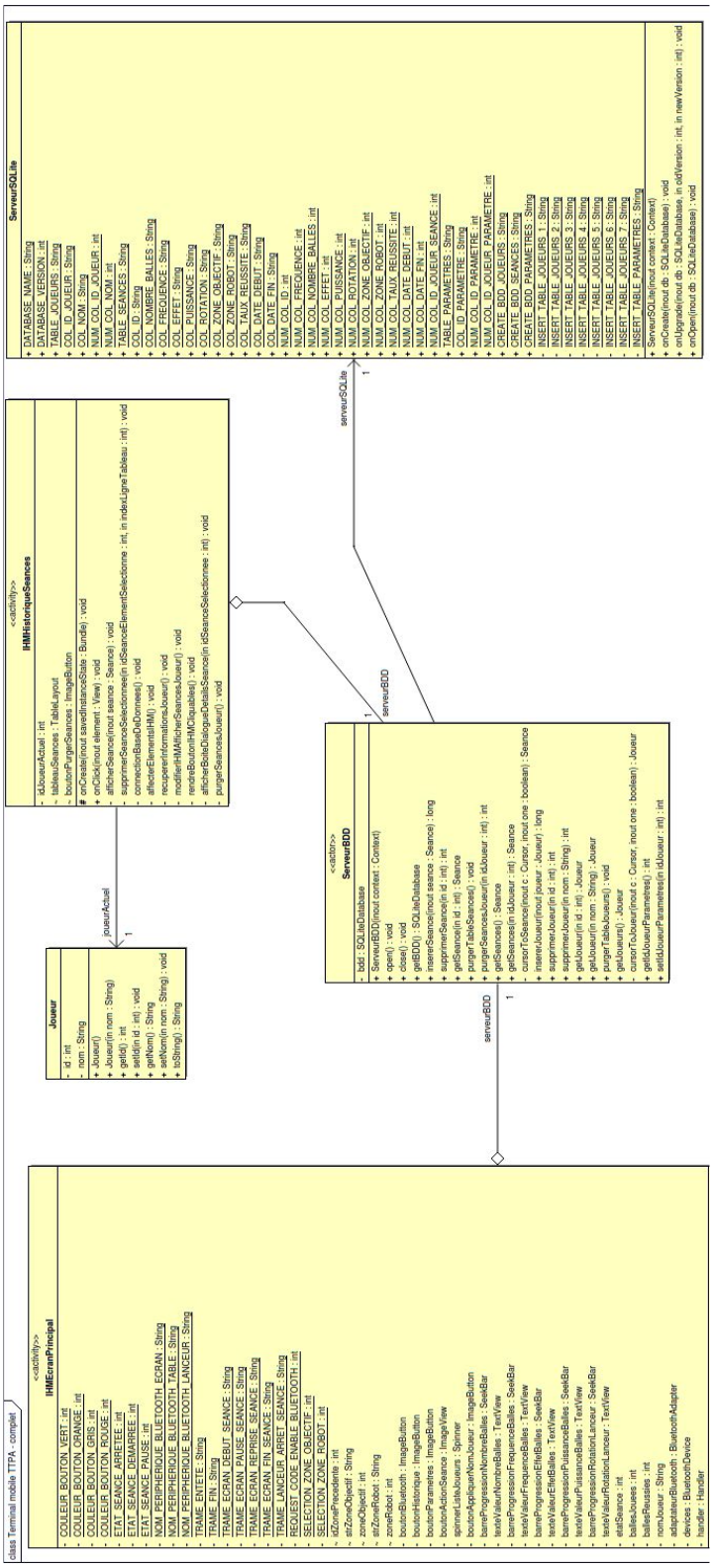
5.7 - Architecture logicielle (globale, non détaillée)



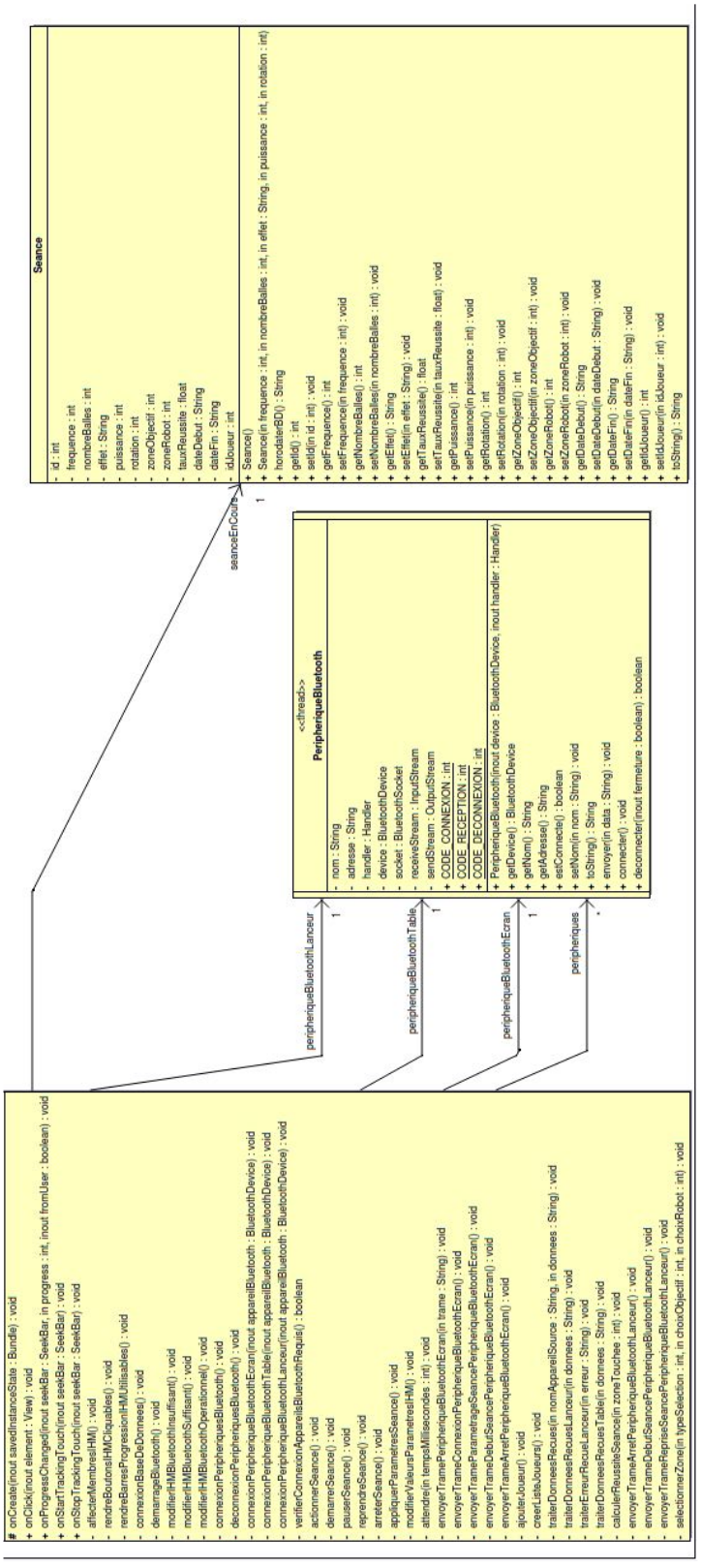
Comme présentées ci-dessus, les classes présentes dans le projet TTPA Terminal Mobile sont les suivantes :

- La classe **AppCompatActivity** est la classe de base fournie par Android décrivant le comportement d'une activité (soit d'une activité comme expliqué précédemment).
- La classe **IHMEcranPrincipale** décrit le comportement de l'activité présentée au démarrage de l'application. Elle hérite donc de la classe **AppCompatActivity**.
- La classe **IHMHistoriqueSeances** est une activité permettant de visualiser l'historique des séances du joueur sélectionné dans l'écran principal.
- La classe **Joueur** décrit les caractéristiques d'un joueur.
- La classe **Seance** décrit les caractéristiques d'une séance.
- La classe **ServeurBDD** va permettre la connexion et l'envoi de requêtes à une base de données **ServeurSQLite**.
- La classe **PeripheriqueBluetooth** permet la communication avec un appareil Bluetooth, dans notre cas, l'envoi et la réception de trames. La réception des trames est assurée par un thread.
- La classe **Handler** est utilisée en tant qu'intermédiaire. En effet, une instance de celle-ci permet de transmettre des messages entre différents fils d'exécution (*thread*) : ici entre le thread de la classe **IHMEcranPrincipale** et le thread de la classe **PeripheriqueBluetooth**.

5.8 - Architecture logicielle (globale, détaillée) *(1/2)*

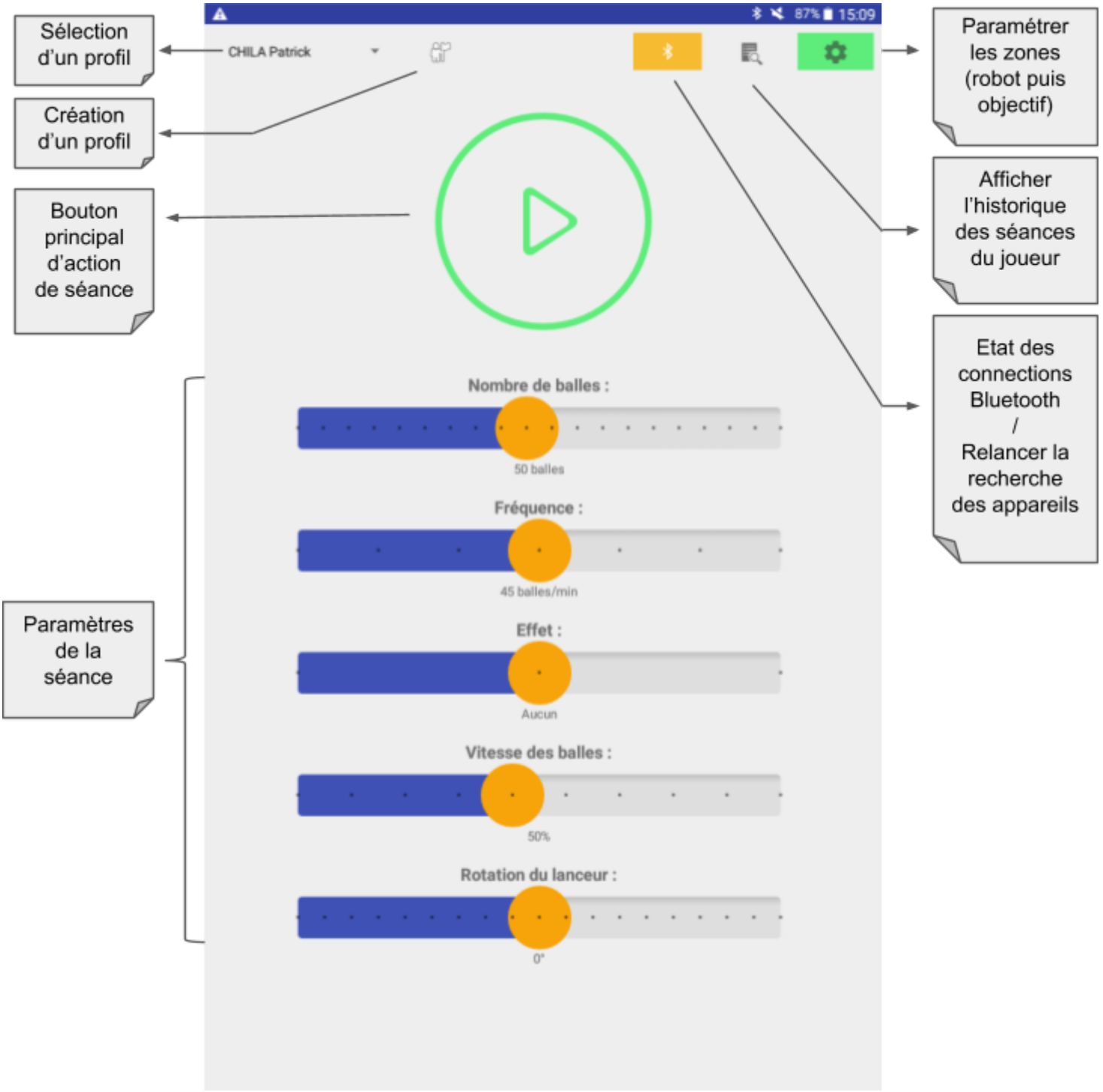


5.9 - Architecture logicielle (globale, détaillée) *(2/2)*

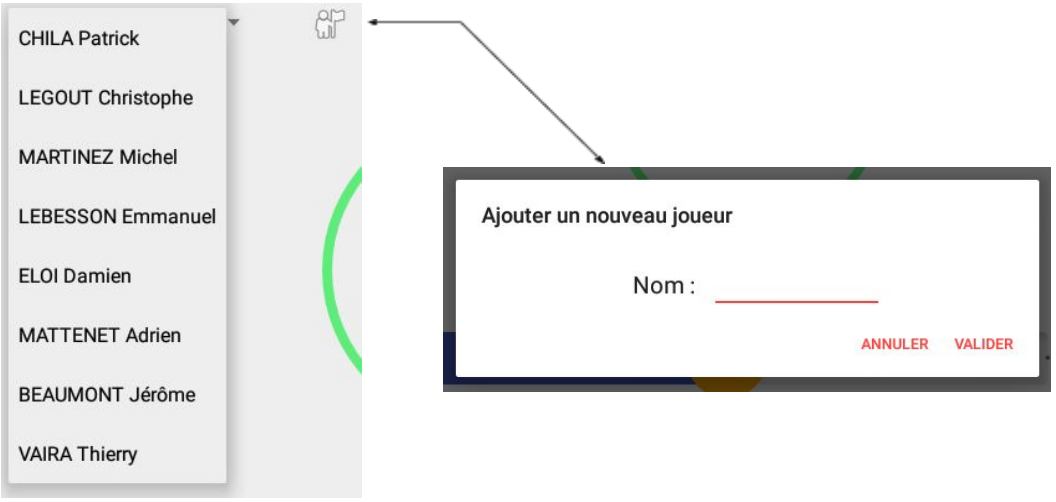


5.10 - Aperçus de l'application

5.10.1 - Au démarrage



C'est l'activité **IHMEcranPrincipal** qui se lance au démarrage de l'application. Cette activité permet au joueur de s'identifier (en choisissant un profil déjà existant ou en en créant un) :



Remarque : le nom est en réalité un pseudonyme permettant d'identifier le joueur, il peut être composé du Nom, Prénom, ou autre.

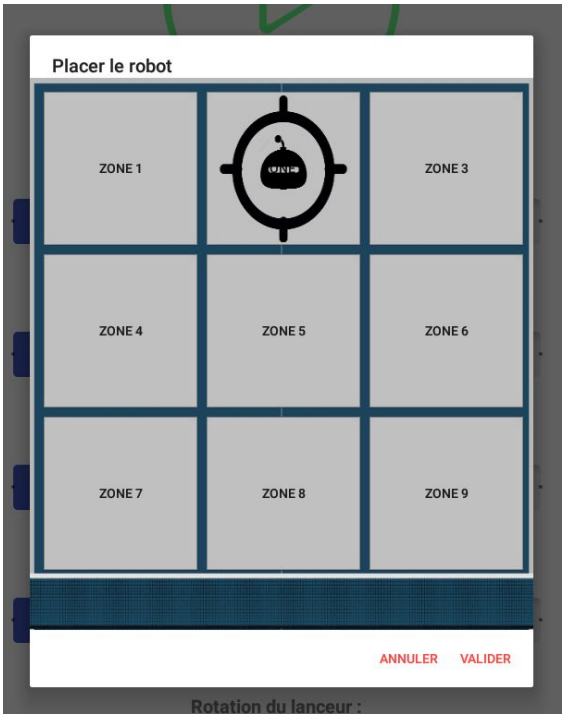
Au démarrage de cette activité, les appareils Bluetooth du système vont être recherchés et selon leur présence, une connexion sera établie.
En fonction des appareils Bluetooth connectés, l'apparence du bouton Bluetooth changera :

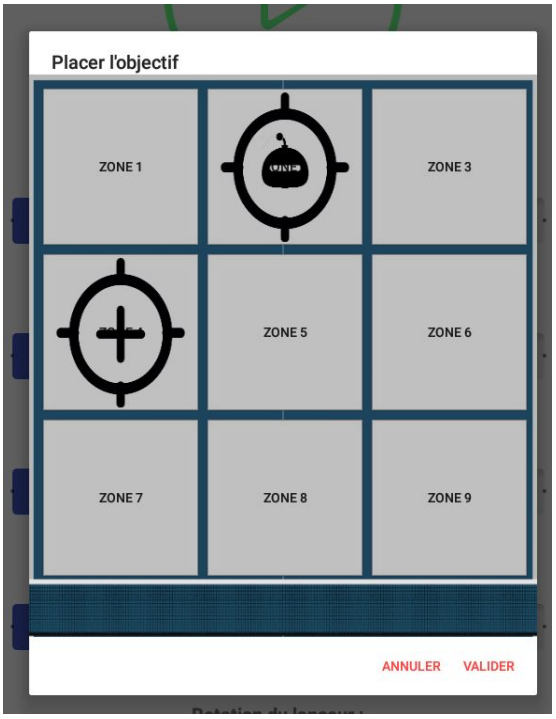
	Apparence initiale, aucune recherche de périphérique Bluetooth n'a été lancée.
	Apparence appliquée lorsqu'aucun appareil du système n'a été detecté.
	Apparence appliquée lorsqu'au moins une connexion à un des appareils Bluetooth a été établie.
	Apparence appliquée lorsque l'application a réussi à se connecter à tous les appareils Bluetooth du système (Table, Écran et Lanceur).

Ce bouton permet la visualisation de l'historique des séances du joueur actuellement sélectionné. En effet, il redirigera l'utilisateur vers une autre activité où les informations des séances précédemment jouée seront présentes.

Lorsqu'aucune connexion à un appareil Bluetooth du système n'est établie, il n'est pas possible de pouvoir régler les zones du robot et de l'objectif.

Lorsqu'au moins une connexion à un appareil Bluetooth est établie, il est désormais possible de débuter une séance, cependant il faudra au préalable placer le robot et l'objectif sur des zones distinctes comme ceci :





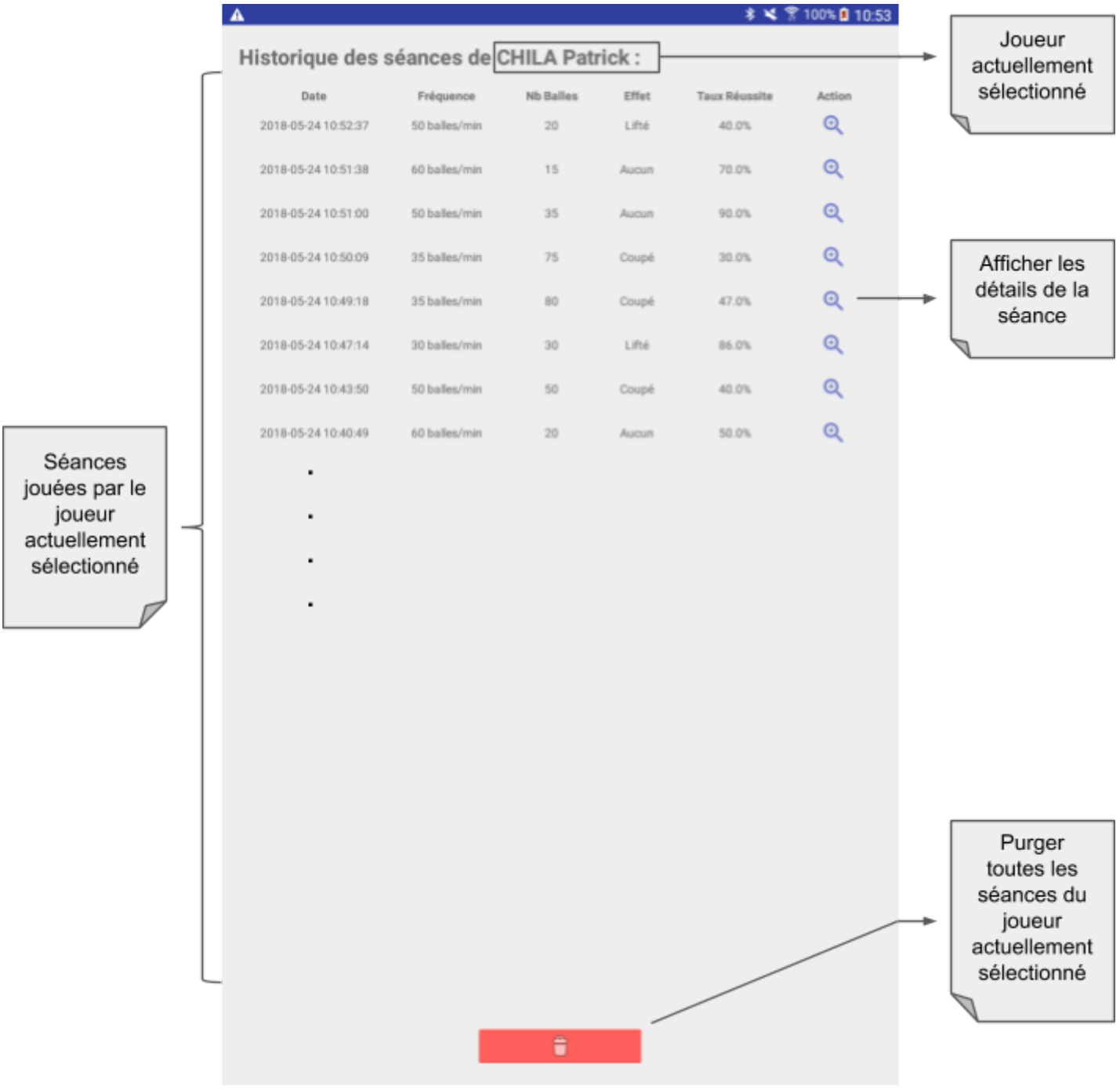
Le bouton principal d'action de séance peut prendre différentes apparences et donc posséder une action à effectuer on conséquent :

Apparence	Etat de la séance	Action à effectuer
	Aucune connexion à un appareil du Bluetooth du système n'est établie, il est donc impossible de démarrer une séance	Aucune
	Au moins une connexion à un appareil Bluetooth du système est établie et aucune séance n'est en cours, il est donc possible d'en démarrer une nouvelle	Démarrer une nouvelle séance avec les paramètres actuels
	Une séance est en cours, il est possible de la mettre en pause	Mettre en pause la séance actuelle
	La séance actuelle est en pause, il est possible de reprendre le cours de celle-ci	Reprendre le cours de la séance actuelle, sans que les paramètres de celle-ci ne soient affectés

Cette activité permet aussi de pouvoir régler, grâce à des curseurs (ou glissières), les paramètres de la séance que l'on veut jouer tels que :

- le nombre de balles à jouer,
- la fréquence d'envoi des balles,
- l'effet de la balle,
- la vitesse de la balle ainsi que la rotation du lanceur.

5.10.2 - Historique des séances du joueur sélectionné

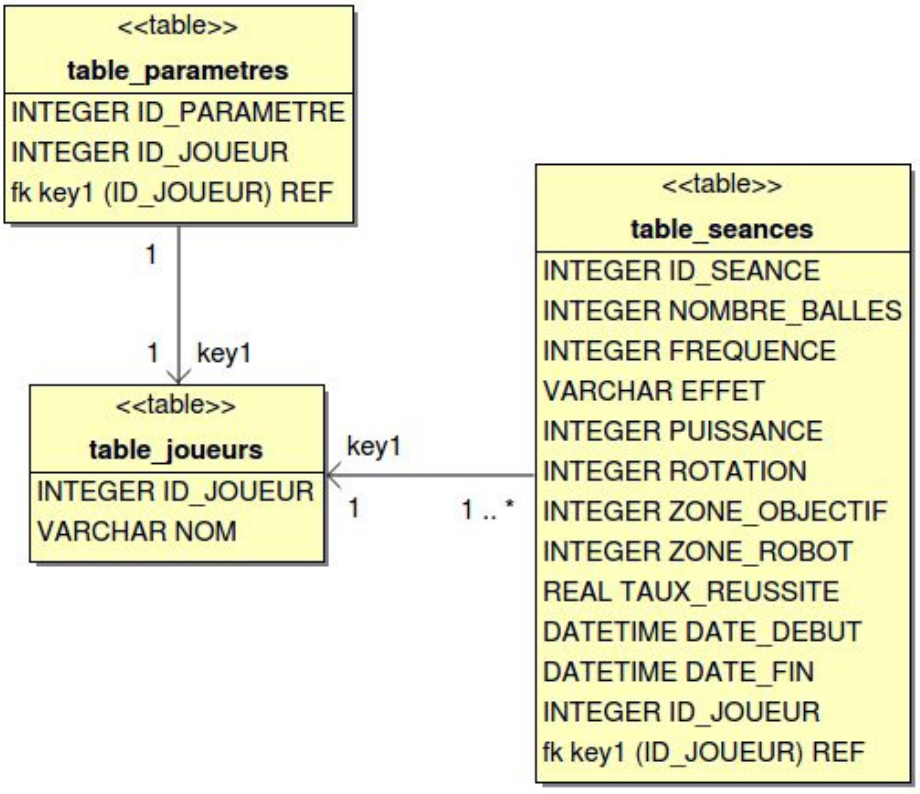


Cette activité, correspondant à la classe **IHMHistoriqueSeances** permet la visualisation des séances enregistrées d'un joueur, comme stipulé dans le cahier des charges.
On peut y apercevoir les paramètres les plus importants d'une séance.
Lorsque le joueur cliquera sur la loupe à droite, il verra apparaître une boîte de dialogue contenant les informations complémentaires de la séance correspondante telles quelles :

Un bouton de purge (suppression) des séances est situé en bas :

La suppression étant irréversible, une boîte de dialogue s'ouvre à l'appui de ce bouton, pour demander à l'utilisateur si il veut confirmer son action :

5.11 - Modélisation de la base de données



Les séances étant enregistrées lorsqu'elles sont terminées (le nombre de balles jouées est égale au nombre de balles paramétré), une entrée est créée dans la base de données pour chaque séance terminée.

Exemple de requête SQL permettant l'insertion d'une nouvelle séance dans la table ***table_seances*** :

```
INSERT INTO table_seances(NOMBRE_BALLES, FREQUENCE, EFFET,
                          PUISSANCE, ROTATION, ZONE_OBJECTIF,
                          ZONE_ROBOT, TAUX_REUSSITE, DATE_FIN,
                          ID_JOUEUR)
VALUES(15, 45, "Aucun", 5, 45, 1 , 3, 40.0, 2018-05-25 15:45:13,
       8);
```

La séance précédemment ajoutée possède ces caractéristiques :

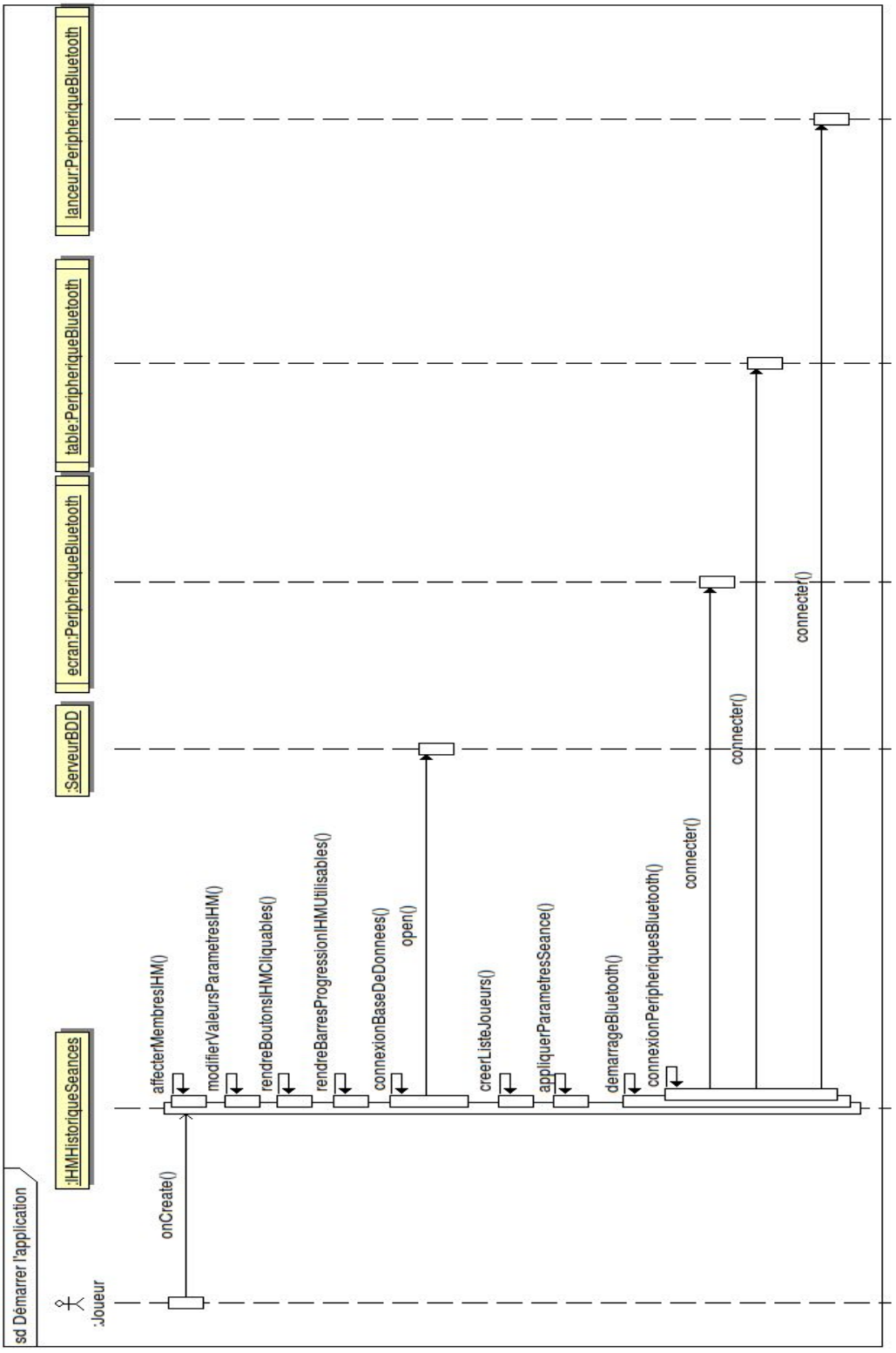
Champ	Valeur
NOMBRE_BALLES	15 (balles)
FREQUENCE	45 (balles/min)
EFFET	“Aucun”
PUISSANCE	5 (coefficient)
ROTATION	45 (°)
ZONE_OBJECTIF	1
ZONE_ROBOT	3
TAUX_REUSSITE	40.0 (%)
DATE_FIN	2018-05-25 15:45:13
ID_JOUEUR	8

Chaque entrée de la table ***table_seances*** est composée d’un champ ***ID_JOUEUR*** qui est une *clé étrangère* faisant référence à la *clé primaire* de la table ***table_joueurs***. Cette relation permet, grâce à la table ***table_parametres*** contenant l’identifiant du joueur actuellement sélectionné, d’afficher uniquement l’historique des séances du joueur sélectionné et non celui des autres joueurs grâce à la requête SQL suivante :

```
SELECT table_seances.NOMBRE_BALLES, table_seances.FREQUENCE,
       table_seances.EFFET, table_seances.PUISSANCE,
       table_seances.ROTATION, table_seances.ZONE_OBJECTIF,
       table_seances.ZONE_ROBOT, table_seances.TAUX_REUSSITE,
       table_seances.DATE_DEBUT, table_seances.DATE_FIN
FROM table_seances, table_parametres
WHERE table_seances.ID_JOUEUR = table_parametres.ID_JOUEUR
ORDER BY ID_SEANCE DESC;
```

5.12 - Scénarios issus du diagramme de cas d'utilisation

5.12.1 - Démarrer l'application



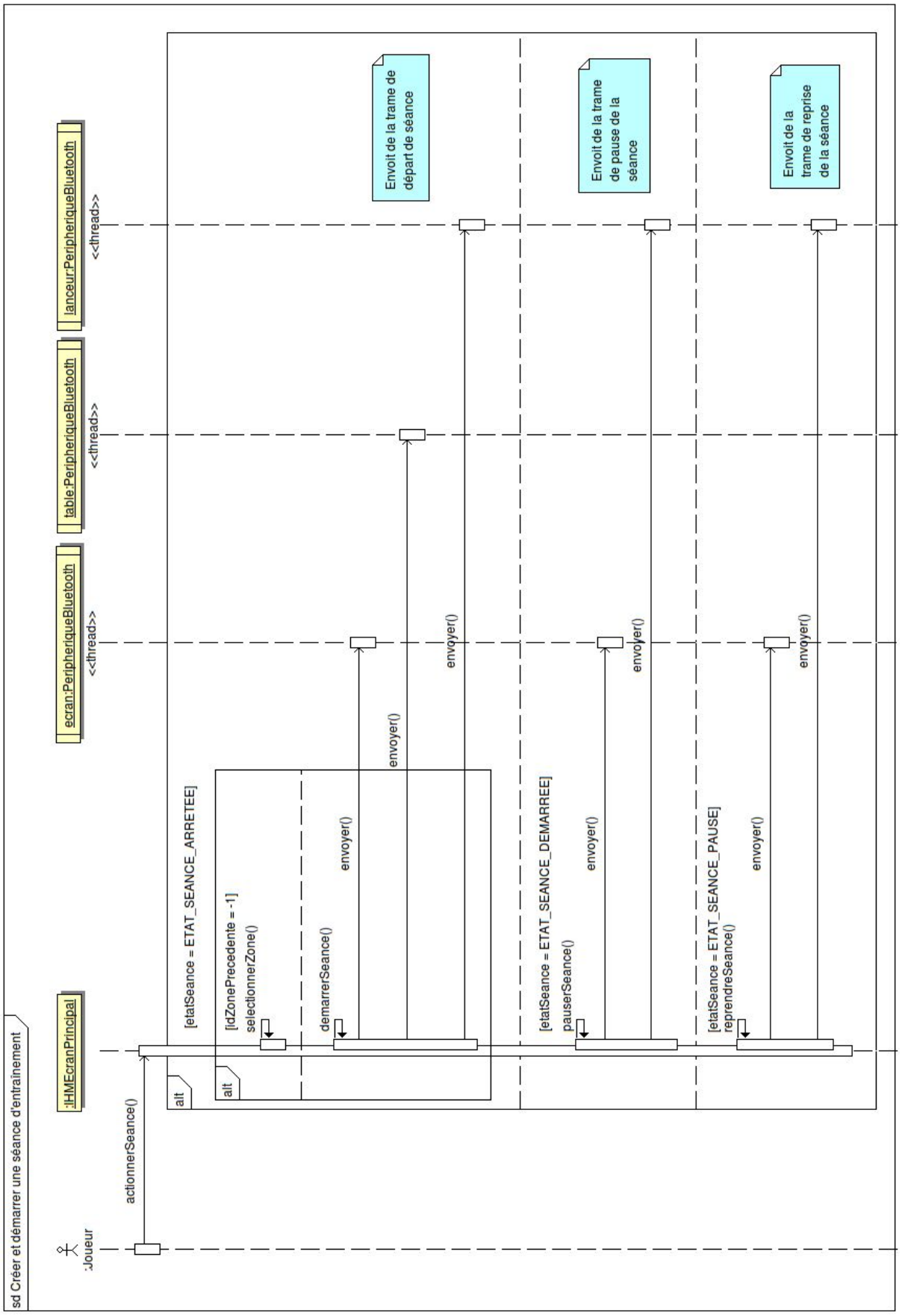
Le diagramme précédent nous décrit les méthodes appelées au démarrage de l'application. En effet, lorsque l'application est ouverte, la méthode `onCreate()` de l'activité **IHMEcranPrincipal** est automatiquement appelée. Celle-ci appellera d'autres méthodes permettant notamment :

- L'initialisation des attributs de la classe ainsi que l'initialisation de l'IHM et ses éléments graphiques.
- La création et la connexion à la base de données.
- La connexion aux appareils Bluetooth disponibles composants le système TTPA.

5.12.2 - Créer et démarrer une séance d'entraînement

Après avoir préalablement ou non paramétré la séance, l'utilisateur doit cliquer sur le bouton principal pour actionner la séance. La méthode appelée (`actionnerSeance()`) va appeler une autre méthode en fonction de l'état actuel de la séance : si elle a déjà été démarrée, si elle est en pause ou arrêtée. Pour chaque action (démarrage, pause, reprise), des trames seront envoyées au différents appareils.

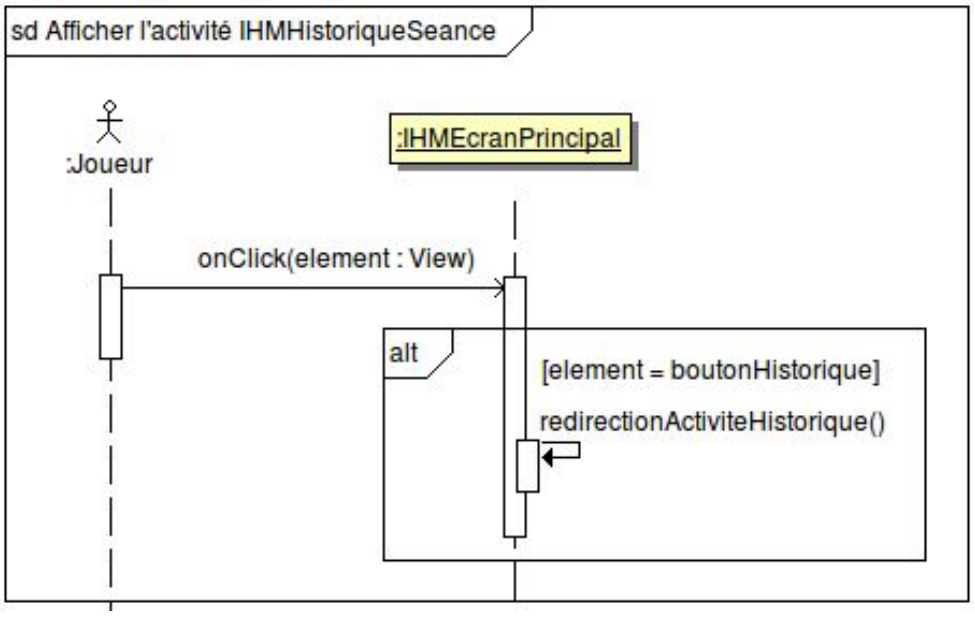
(voir diagramme de séquence ci-dessous)


sd Créer et démarrer une séance d'entrainement
:Joueur
:IHMEcranPrincipal
ecran:PeripheriqueBluetooth
<<thread>>
table:PeripheriqueBluetooth
<<thread>>
lanceur:PeripheriqueBluetooth
<<thread>>
actionnerSeance()
alt
[etatSeance = ETAT_SEANCE_ARRETEE]
[idZonePrecedente = -1]
selectionnerZone()
alt
demarrerSeance()
envoyer()
envoyer()
envoyer()
Envoit de la trame de départ de séance
[etatSeance = ETAT_SEANCE_DEMARREE]
pauserSeance()
envoyer()
envoyer()
Envoit de la trame de pause de la séance
[etatSeance = ETAT_SEANCE_PAUSE]
reprendreSeance()
envoyer()
envoyer()
Envoit de la trame de reprise de la séance

5.12.3 - Consulter l'historique des séances

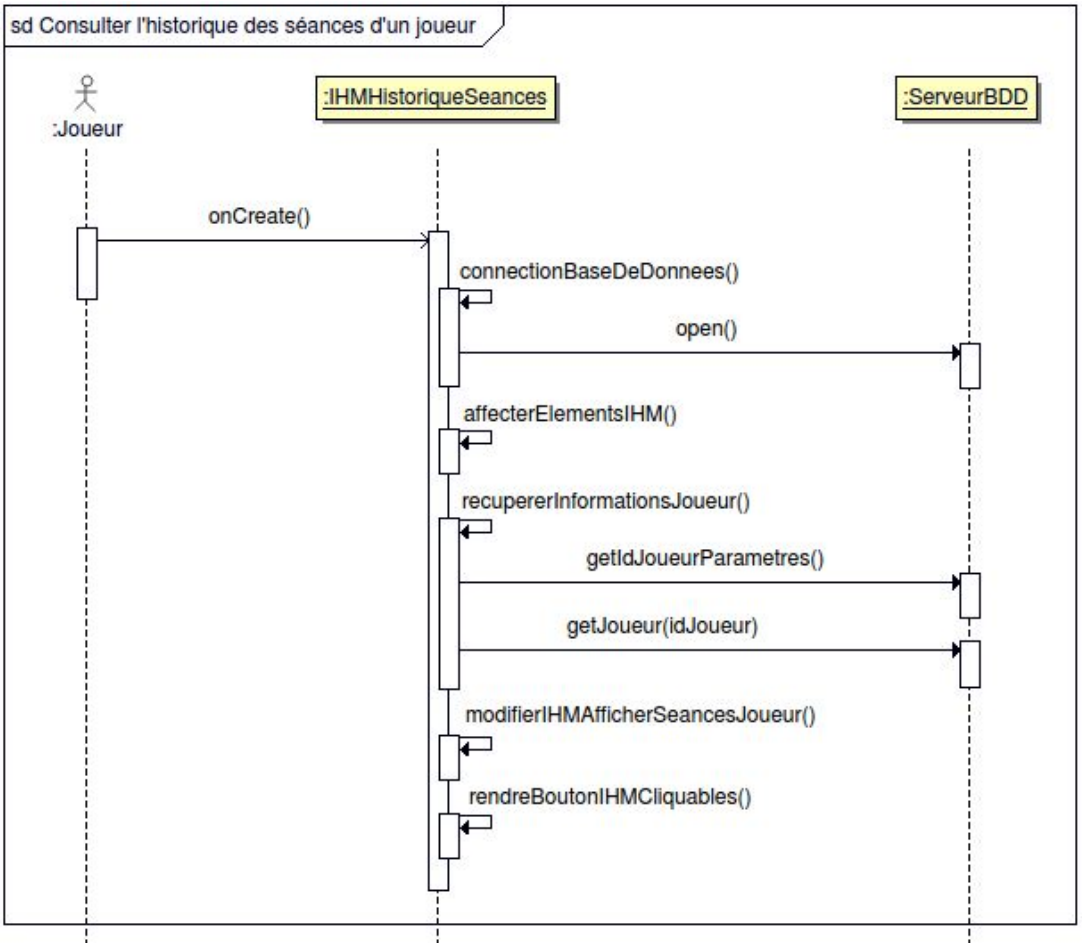
Avant de pouvoir visualiser son historique de séances, le joueur doit cliquer sur le bouton d'historique :

Lorsqu'il est cliqué, ce bouton appelle une méthode **`redirectionActiviteHistorique()`** comme ceci :




```
/**
* Déclaration de la méthode redirectionActiviteHistorique démarrant
l'activité IHMHistoriqueSeances permettant de visualiser l'historique des
séances du joueur actuel.
*/
private void redirectionActiviteHistorique()
{
   Log.d("IHMEcranPrincipal", "redirectionActiviteHistorique()");

   Intent intent = new Intent(IHMEcranPrincipal.this, IHMHistoriqueSeances.class);
   startActivity(intent);
}
```



L'activité **IHMHistoriqueSeances** est donc affichée, appelant sa méthode `onCreate()` qui va permettre notamment d'initialiser ses composants graphiques, puis d'envoyer des requêtes à la base de données afin :

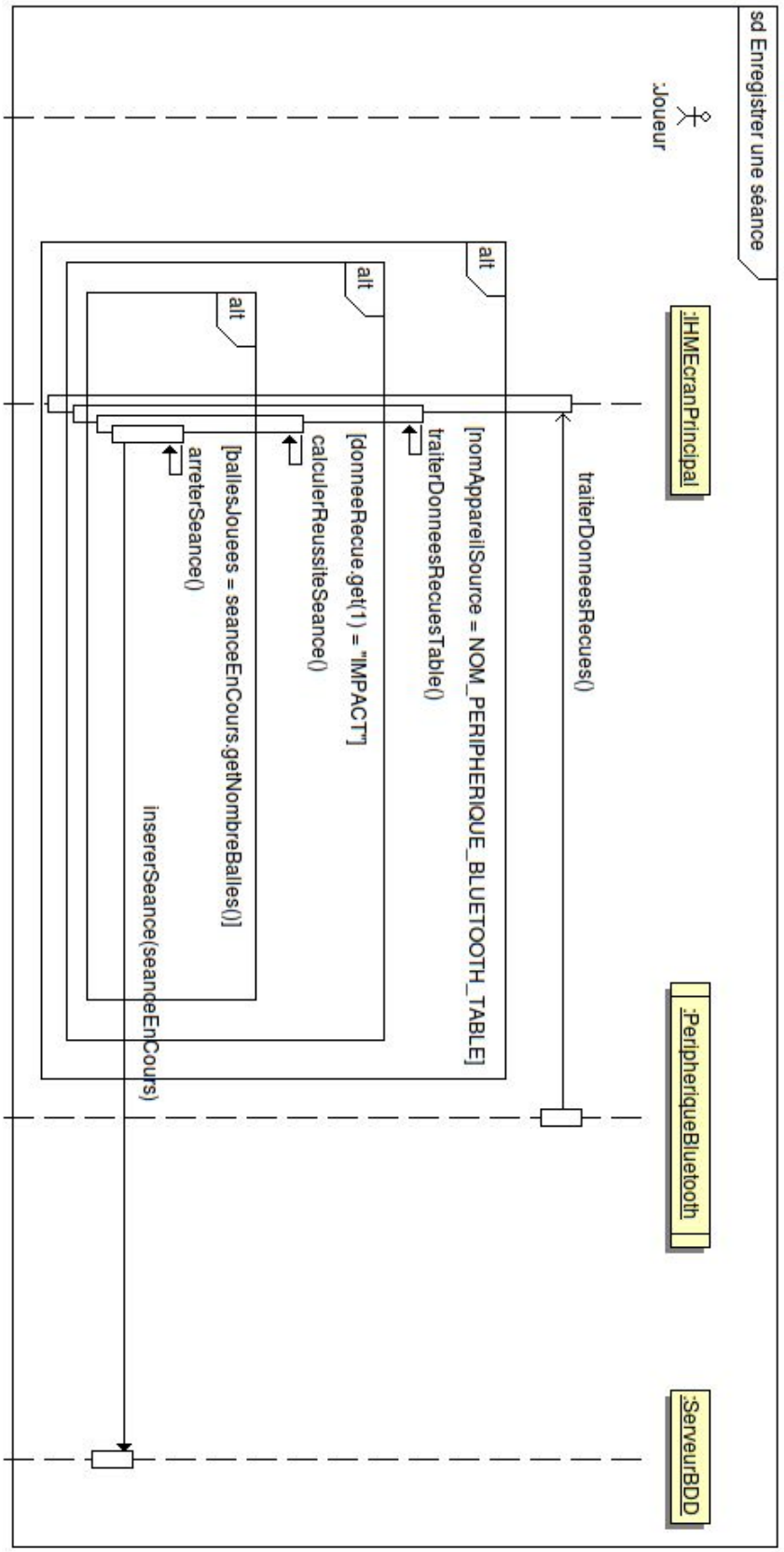
- d'acquérir l'identifiant du joueur actuellement sélectionné, que l'on va stocker dans une variable, grâce à la requête SQL suivante :

```
SELECT table_joueurs.ID_JOUEUR, table_joueurs.NOM
FROM table_joueurs, table_parametres
WHERE table_parametres.ID_PARAMETRE = '1'
AND table_joueurs.ID_JOUEUR = table_parametres.ID_JOUEUR;
```

- d'afficher les séances jouées par celui-ci grâce à la requête SQL suivante :

```
SELECT *
FROM table_seances
WHERE ID_JOUEUR = idJoueurActuel;
/* idJoueurActuel étant une variable stockant l'id du joueur
récupéré précédemment */
```

5.12.4 - Enregistrer une séance



La visualisation des séances jouées nécessite qu'elles soient enregistrées. En effet, les séances sont enregistrées lorsqu'elles sont terminées. La détermination de la fin de la séance se fait ainsi :
Lorsque l'on reçoit une trame en provenance de la table (système de détection d'impact), elle est traitée.
En effet, si elle correspond à une trame d'impact, cela signifie qu'une balle a touché une certaine zone de la table.
Grâce à la zone reçue en paramètre de la trame, le taux de réussite de la séance est calculé selon la zone d'objectif placée (ou non), incrémentant le nombre de balles qui ont été jouées pour la séance actuelle.
Enfin lorsque le nombre de balles jouées pour la séance actuelle correspond au nombre de balles à jouer paramétré en début de séance, cela signifie que celle-ci est terminée.
Une méthode **`arreterSeance()`** de la classe **IHMEcranPrincipal** est donc appelée.
Celle-ci va entre autre communiquer la fin de séance aux différents appareils Bluetooth du système, mais aussi enregistrer les caractéristiques de la séance grâce à la méthode **`insererSeance(seanceEnCours)`** de la classe **ServeurBDD**, en passant en argument un objet de classe Seance. Voici le code source de cette méthode :

```
/**
* Déclaration de la méthode insererSeance permettant l'insertion d'une séance dans
la base de données.
* @param seance étant la séance à insérer dans la base de données.
* @return l'id de la séance dans la base de données.
*/
public long insererSeance(Seance seance)
{
   ContentValues values = new ContentValues();

   values.put(ServeurSQLite.COL_FREQUENCE, seance.getFrequence());
   values.put(ServeurSQLite.COL_NOMBRE_BALLES, seance.getNombreBalles());
   values.put(ServeurSQLite.COL_EFFET, seance.getEffet());
   values.put(ServeurSQLite.COL_PUISSANCE, seance.getPuissance());
   values.put(ServeurSQLite.COL_ROTATION, seance.getRotation());
   values.put(ServeurSQLite.COL_ZONE_OBJECTIF, seance.getZoneObjectif());
   values.put(ServeurSQLite.COL_ZONE_ROBOT, seance.getZoneRobot());
   values.put(ServeurSQLite.COL_TAUX_REUSSITE, seance.getTauxReussite());
   values.put(ServeurSQLite.COL_DATE_DEBUT, seance.getDateDebut());
   values.put(ServeurSQLite.COL_DATE_FIN, seance.getDateFin());
   values.put(ServeurSQLite.COL_ID_JOUEUR, seance.getIdJoueur());

   return bdd.insert(ServeurSQLite.TABLE_SEANCES, null, values);
}
```

Cette méthode permet d'insérer un enregistrement d'une séance dans la table **table_seances** de la base de données, en utilisant les attributs de l'objet de classe **Seance** passé en argument.

5.13 - Tests de validation (recette finale)

Description	OUI	NON
La base de données est fonctionnelle et complétée	✓	
Le système est paramétrable	✓	
La liaison Bluetooth est fonctionnelle	✓	
Les informations de paramétrages sont transmises	✓	
L'application mobile est déployée	✓	

Les objectifs donnés par le cahier des charges ont été entièrement réalisés. Cependant, il reste des améliorations graphiques et techniques possibles.

6 - Conclusion

6.1 - État de l'avancement

Les objectifs de base ont été réalisés et l'application est déployée.
La partie Terminal Mobile est désormais fonctionnelle seule.
De plus, elle a été testée avec des simulateurs lors du développement puis avec les appareils réels du système et les essais des fonctionnalités ont été concluants.
Il reste cependant quelques améliorations :

- Améliorations graphiques afin que l'application possède le même visuel sur n'importe quelle plateforme (smartphone, tablette, etc …),
- Pouvoir mieux visualiser l'état des appareils Bluetooth pour savoir avec lesquels la connexion a été établie,
- Rafraîchir les états des appareils Bluetooth pour gérer les déconnexions non prévues,
- Pouvoir régler l'intensité de l'effet, avec des paliers,
- Pouvoir supprimer individuellement une séance,
- Afficher toutes les informations dans les détails d'une séance :
    - durée de la séance,
    - vitesse des balles,
    - intensité de l'effet,
    - rotation du lanceur
- Calculer la durée de la séance (en prenant en compte les pauses),
- Ajouter un bouton *RETOUR* sur l'activité **IHMHistoriqueSeances** afin de revenir à l'écran principal.

6.2 - Bilan global

Ce projet m'a tout d'abord apporté une expérience ainsi que la maîtrise de l'environnement de développement **Android Studio** et du langage de programmation **Java**. Il m'a aussi permit de pouvoir approcher les attentes d'un projet en milieu professionnel, telles que : la communication au sein d'une l'équipe, le respect des consignes initiales du client ainsi le respect de contraintes dans le temps.

Adrien RACAMOND

BTS SN - Session 2018

Table Tennis Performance Analyser

ECRAN-TTPA v1.0

Revue Finale

campus La Salle

SOMMAIRE

Besoin Initial

Ce système doit fournir une analyse de performance d'un joueur de tennis de table face à un robot lanceur pendant une séance d'entraînement. Le joueur a pour objectif de relancer la balle dans une "zone" précise de la table. La table est découpée en 9 "zones", cette précision est donc limitée à ces cases. La performance du joueur pendant la séance est basée sur le nombre de balles renvoyées dans la zone à atteindre. Le robot lanceur est paramétrable afin d'obtenir une variété de balles réaliste. Ce projet est en partenariat avec: Ping Pong Club Sorgues, basé sur un besoin réel du club pour leurs entraînements.

Objectifs

Le système doit être capable de proposer une séance d'entraînement adapté au niveau du joueur.

Pour cela, il est décomposé en 4 modules :

- **Module de paramétrage de séance (Mobile-TTPA)** : le joueur et/ou l'entraîneur paramètre et lance la séance d'entraînement à partir d'une application sur un terminal mobile (sous Android) ;
- **Module de distribution de balles (Robot-TTPA)** : le robot lanceur envoie des balles, en suivant un rythme fixé par le joueur, avec des effets réalistes et différents ;
- **Module de détection des impacts (Table-TTPA)** : la table est équipé de 9 capteurs permettant d'identifier la zone impactée par la balle renvoyée par le joueur ;
- **Module de visualisation de performance (ECRAN-TTPA)** : l'entraîneur peut visualiser en "temps réel" le rythme de jeu, la précision, le pourcentage de réussite.

Sur le terminal mobile, l'application doit permettre de créer et démarrer une séance d'entraînement.

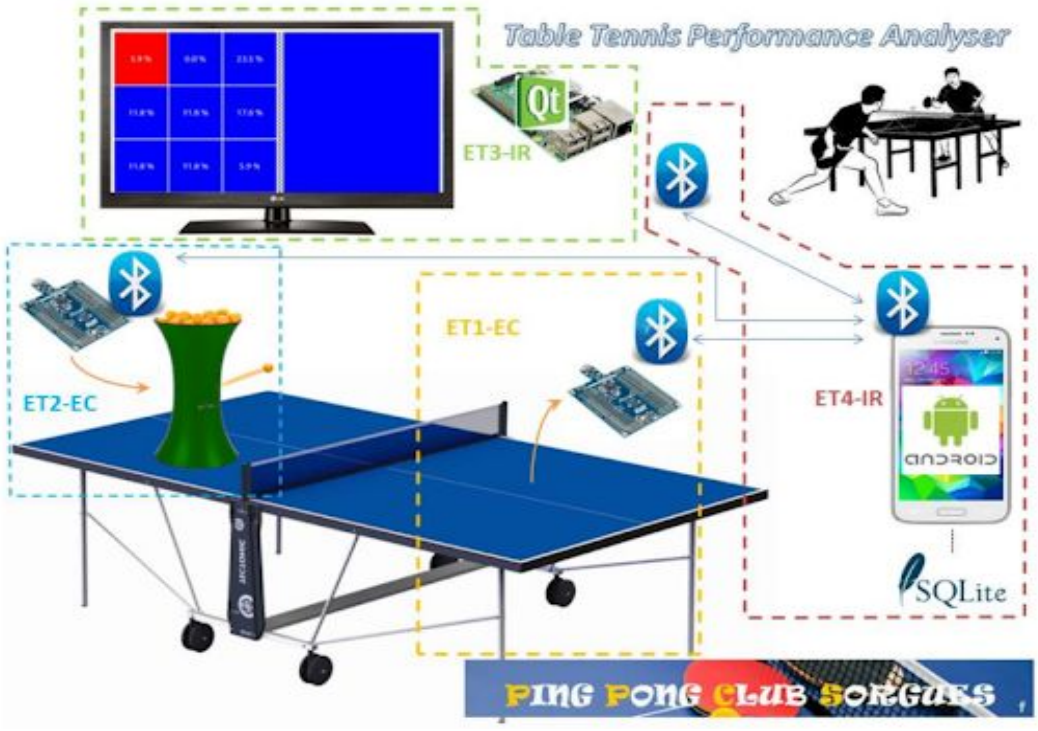
Pour cela, le joueur ou l'entraîneur pourra :

- saisir son nom
- paramétrer le lanceur de balles :
    - le nombre de balles à jouer pour la séance,
    - le délai d'envoi entre chaque balle,
    - les effets et la puissance des balles
- déterminer la position du robot et la zone à atteindre sur la table

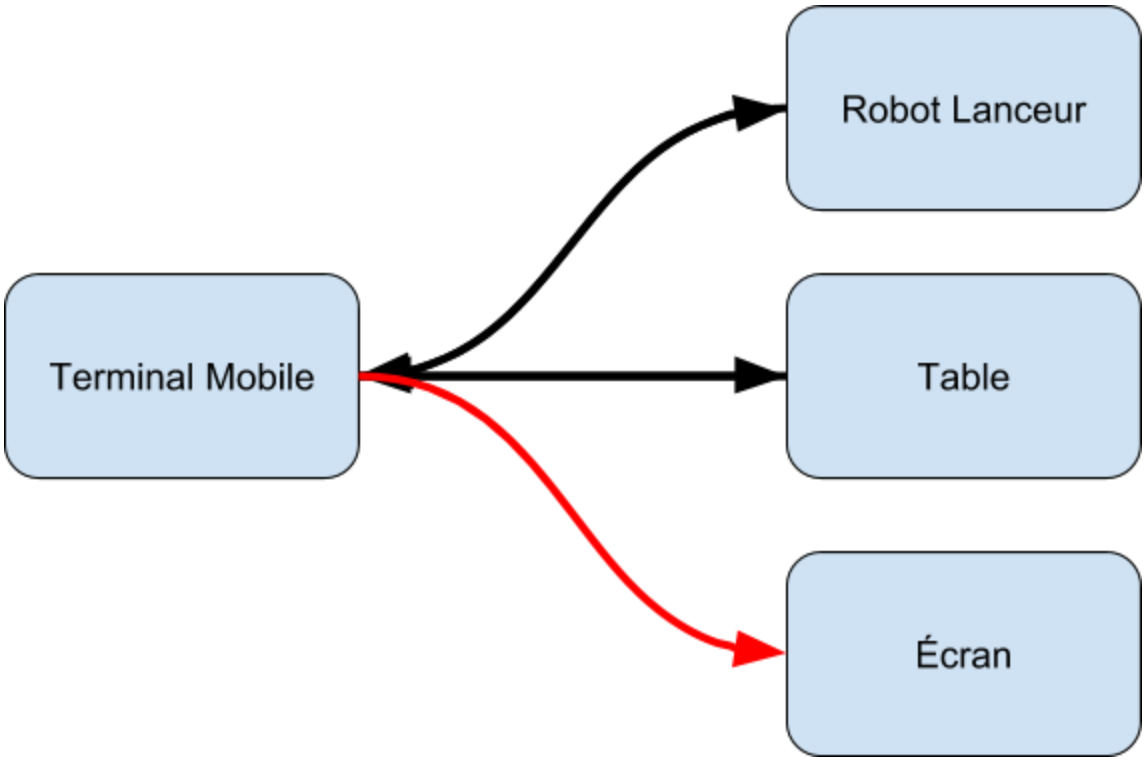
Le terminal mobile possède une base de données permettant de stocker les statistiques du joueur de chaque séance d'entraînement.

Sur l'écran, l'entraîneur pourra visualiser en continu :

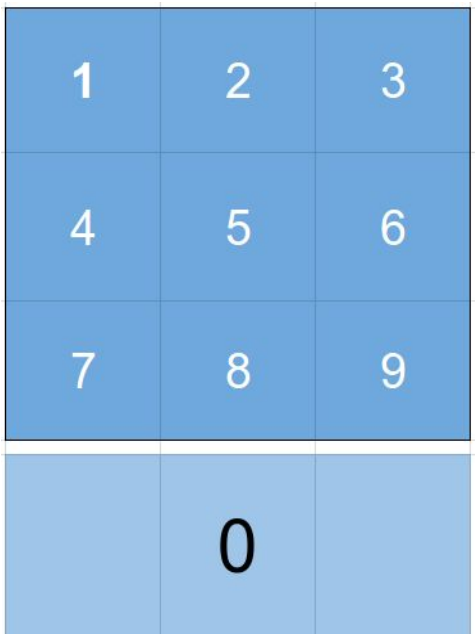
- le nom du joueur (si existant), la durée écoulée de la séance
- le nombre de balles à jouer, le nombre de balles jouées, le nombre total de coups dans la zone à atteindre
- l'ensemble des zones avec le pourcentage de coups atteints pour chacune
- la zone à atteindre (en vert par exemple) et la zone où le coup a été joué (en rouge par exemple)



Les modules communiquent en Bluetooth :



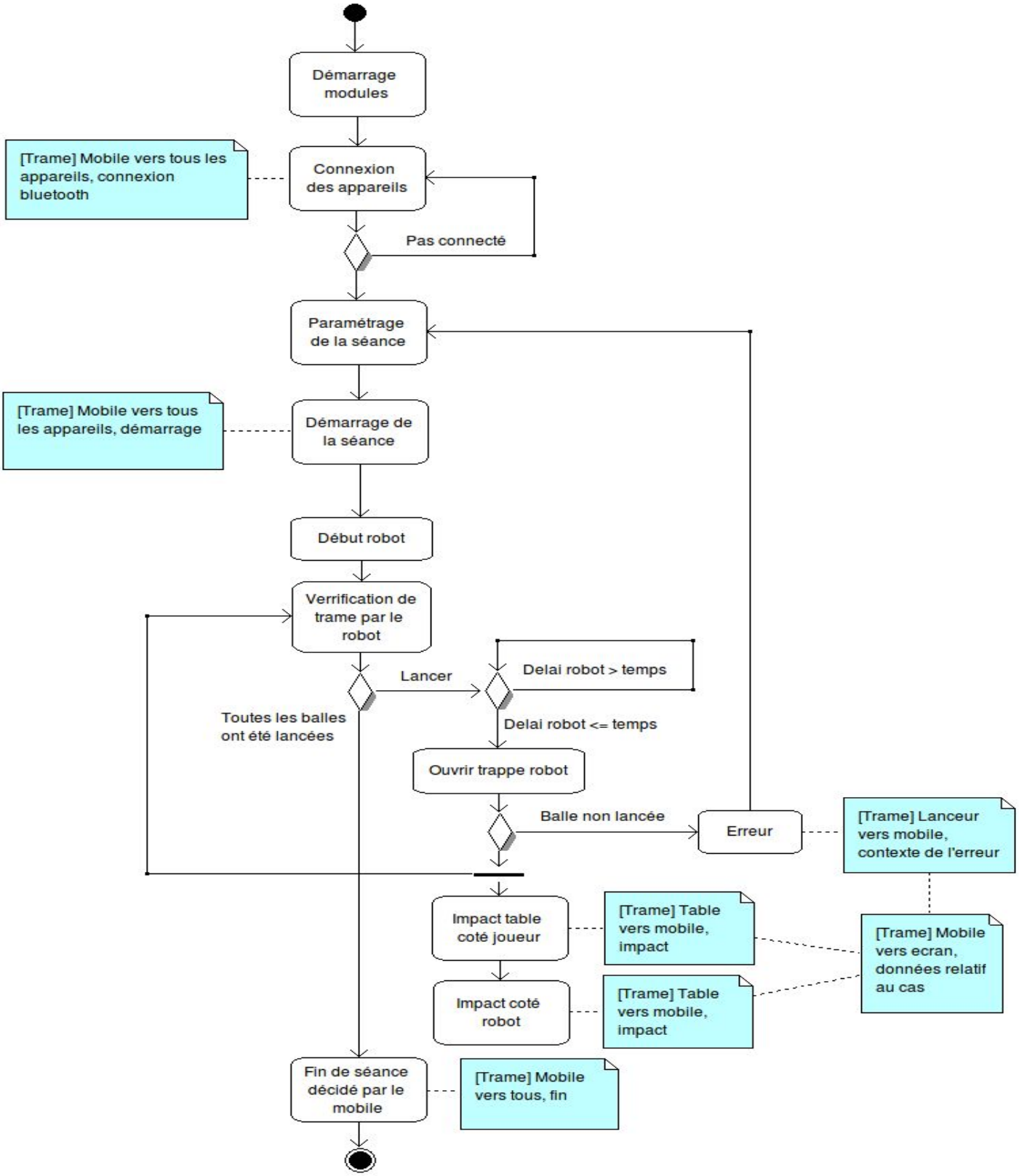
Les différentes “zones” d’une table sont :



- Le **haut** est la partie de la table comptabilisant les balles dans les zones pour les statistiques, il s’agit du côté où le **robot lanceur** est placé (celui ci peut aussi être désactivé).

- Le **bas** est le côté du **joueur**, ou seul l’impact de balles est détecté à des fins de vérifications (permet de vérifier si la balle envoyé par le robot peut être joué par le joueur).

Diagramme d'activité du système TTPA



Le sous-système ECRAN-TTPA

Introduction

Le sous-système ECRAN-TTPA (Module de visualisation de performance) correspond à la partie “affichage” du système. Il a pour objectifs de réaliser la **récupération d’informations** envoyées par le terminal mobile, **le calcul et l’affichage les statistiques** pour la séance actuelle.

Diagramme des cas d’utilisation

L’**entraîneur** peut visualiser en continu les données (pourcentage de balles par zones et le nombre de balles dans la zone à atteindre) d’une séance d’entraînement. Pour cela, le sous-système a besoin de récupérer les trames émises par le terminal mobile.

Les données visualisées sont :

- Le pourcentage de balles dans chaque zone en comptant les balles hors table
- Le nombre et pourcentage de balles hors table (une balle n’ayant pas touché une zone)
- Le nombre de balles jouées par le robot sur le nombre de balles pour la séance
- L’heure ainsi que le temps de la séance depuis son lancement
- Le nom du joueur

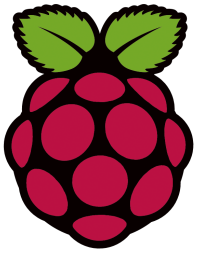
Une communication entre le sous-système ECRAN-TTPA et le terminal mobile basée sur un protocole spécifique est nécessaire.

Ressources matérielles

- PC pour le développement du programme, sous Linux Ubuntu 12.04 LTS
- Raspberry Pi 3 pour son exécution, sous Raspbian 9.1.

Ressources logicielles

- Qt 4.8.1
- Planner 0.14.5
- BOUML 7.4
- Doxygen 1.7.6.1
- Subversion 1.6.17



L'application Ecran-TTPA est écrite en C++ utilisant l'API Qt et compilée pour un système Raspberry Pi (ARM). L'application s'exécute en mode "Kiosque" (*kiosk*). En effet, la Raspberry Pi est dédiée à l'exécution unique de l'application ECRAN-TTPA qui doit se lancer automatiquement à la mise sous tension.

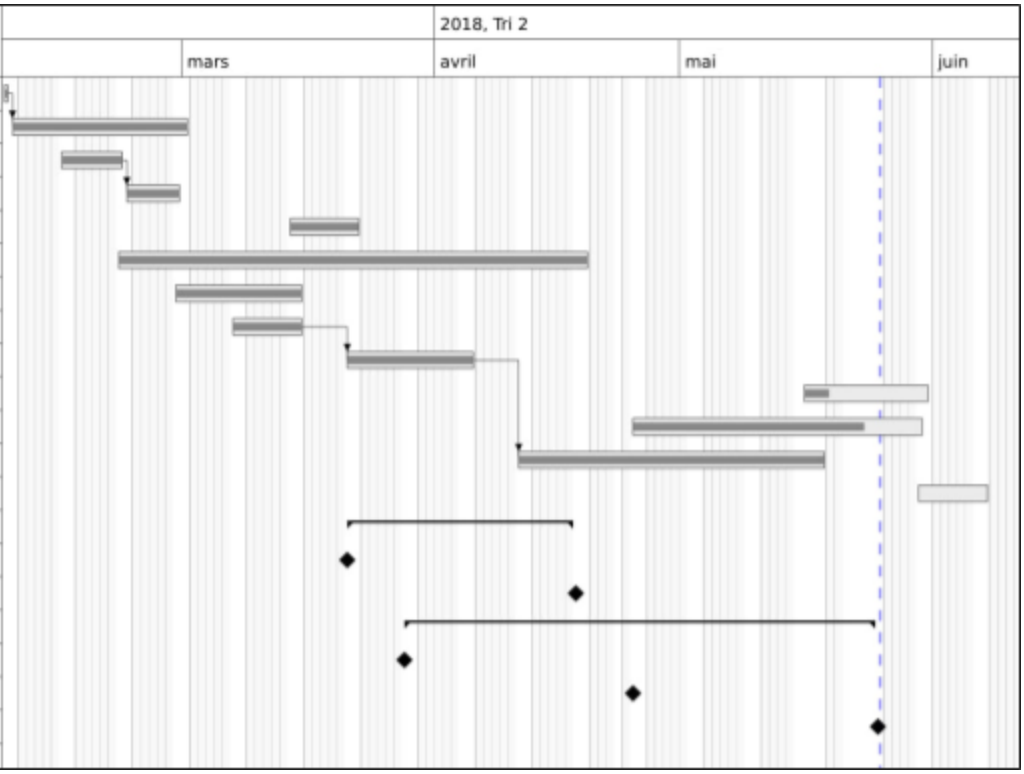
Le mode "Kiosque" (*kiosk*) a été conçu à l'origine pour les bornes internet (*Internet kiosks*). Ces bornes nécessitent la mise en place d'un environnement logiciel restreint aux fonctions essentielles afin d'empêcher les utilisateurs de faire quoi que ce soit d'indésirable ou de dangereux. Cela comprend aussi :

- de dédier l'affichage graphique exclusivement à une application en plein écran
- d'enlever les composants graphiques habituels (curseur souris, barre de titre, tableau de bord, icônes, etc ...)
- de désactiver les modes économiseur d'écran (notamment l'écran de veille)

Ce mode est particulièrement utile pour les afficheurs autonomes, comme des bornes accessibles au public. Parmi les exemples les plus couramment rencontrés : les distributeurs de billets, les affichages d'horaires dans les gares, ...

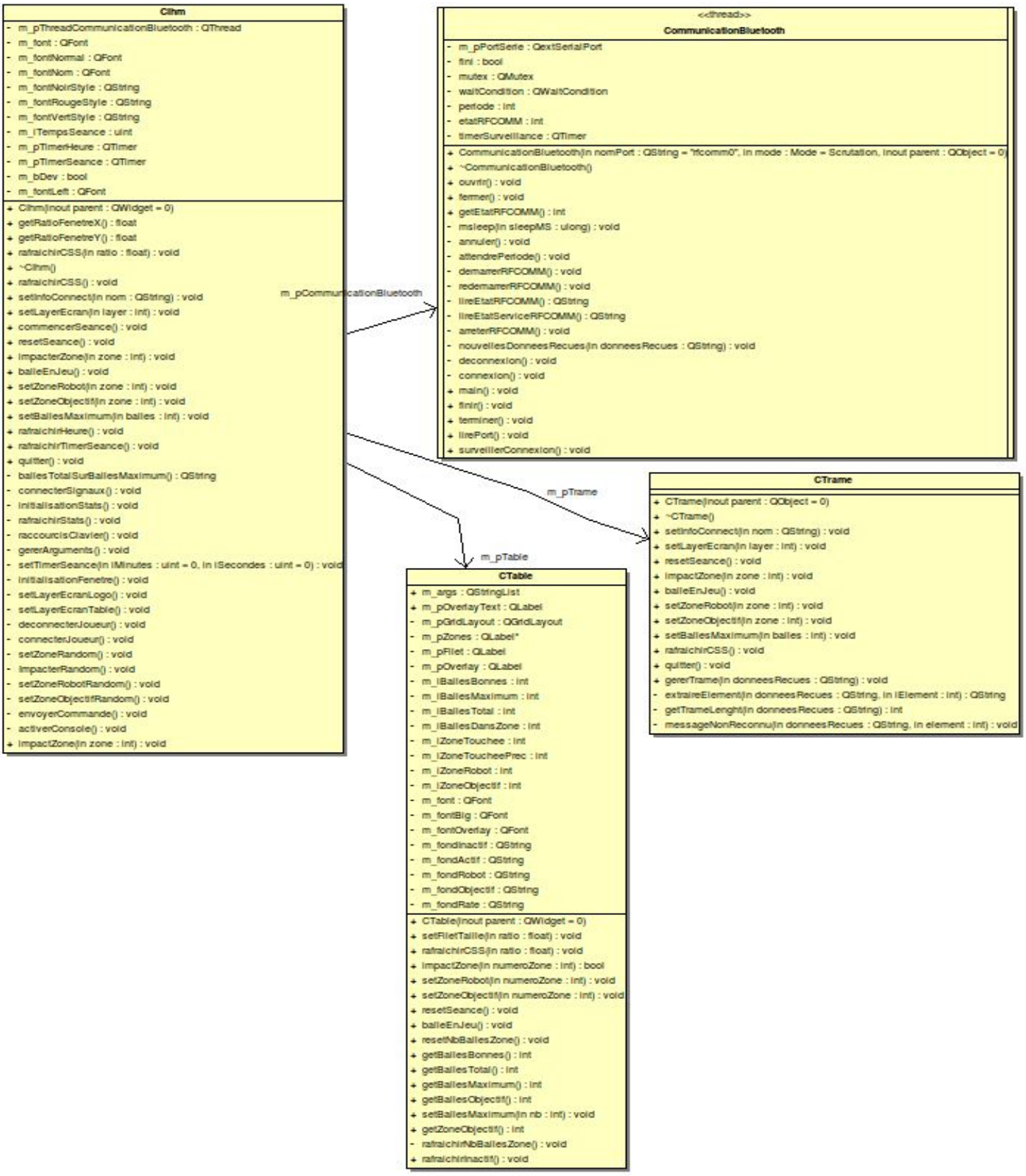
Planification

Nom	Travail
Planification	1j
Coder base Interface et Table	7j
Coder statistiques Zones %	3j
Ajouter Robot + Zone Objectif	2j
Coder base Ecran d'acceuil	4j
Developper Interface	18j
Ajouter Bluetooth	6j
Protocole	4j
Developper Comm BT / Ecran	6j
Ajout de fonctionalités	9j 3h
Ajout de statistiques récapitulatifs	16j
Redaction intensive du dossier	15j
Préparation demonstration	4j
{Vérifications}	
Vérifier + Reorganiser Code	
Vérifier Protocole	
{Revues}	
Revue	
Revue	
Rendu du dossier	



- Le développement de l'interface de la table était le plus logique a faire en premier. Il s'agit de l'élément le plus important et le plus visible. Les dimensions de la fenêtre du programme ont été "pensées" pour un écran 16/9. Pour faciliter le développement ultérieure, on a besoin de fonctions pour redimensionner l'interface en fonction de la résolution de l'écran (contraintes de taille de police, d'échelle des éléments, de la taille du filet etc...).
- Le développement a d'abord fonctionné sans interface utilisable, c'est par la suite que l'option de lancement **`-dev`** a été mis en place pour tester du fait que le système est seulement prévu de contrôler le système à distance par Bluetooth.
- Par la suite les pourcentages par zones ont été ajouté ainsi que la prise en compte du robot et de la zone objectif sur la table, le tout utilisant des QLabel avec des couleurs définis en CSS en fonction de leur état.
- L'ajout du Bluetooth, création du protocole et l'ajout des fonctions pour la communication entre le terminal mobile n'est venu qu'après.
- et enfin l'ajout de la fenêtre récapitulative vers la fin du projet.

Diagramme de classes



La classe principale est **`CIhm`**. Elle a en charge l'affichage des données à l'écran. Elle confie l'affichage de la table à la classe **`CTable`**.

- La classe **`CTable`** gère la partie table de l'interface ainsi que les calculs lié aux pourcentages de balles dans les zones, balles maximum de la séance et les balles jouées par le robot. (voir page 23)
- La classe **`CommunicationBluetooth`** gère la communication avec le port virtuel **`rfcomm`**, récupère les trames du terminal mobile, et les envoie par événement (notion de *signal* Qt). (voir page 12)
- La classe **`CTrame`** s'occupe du découpage des trames afin de récupérer les paramètres pour les fournir aux fonctions de la classe **`CIhm`** (voir page 16).

Specifications du Raspberry Pi 3

Processeur	Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
RAM	1GB RAM
Stockage	Micro SD
Sans-Fil	BCM43438 LAN et "Bluetooth Low Energy" (BLE)
Connectiques	40-pin GPIO, 4 ports USB 2, HDMI, CSI, DSI
Modèle	Raspberry Pi 3 Model B Rev 1.2
OS	Raspbian GNU/Linux 9.1 (stretch)

Mise en oeuvre du Bluetooth sur la Raspberry Pi 3

Dans ce système, le port **`rfcomm`** (*Radio Frequency Communication*) est utilisé comme passerelle vers le périphérique Bluetooth interne de la Raspberry Pi.

L'installation de **`bluetoothctl`** et d'autres programmes sont importants :

```
sudo apt-get install bluetooth bluez bluez-tools rfkill rfcomm
```

Il n'est pas nécessaire de démarrer les services bluetooth, le programme ECRAN-TTPA vérifie son état au démarrage, et s'occupe de l'activer, il s'agit de l'un des rôles de la classe active **`CommunicationBluetooth`** :

```
void CommunicationBluetooth::demarrerRFCOMM()
{
    FILE *resultat;
    resultat = popen("sudo systemctl start rfcomm.service 2> /dev/null", "r");
    pclose(resultat);
}
```

`rfcomm.service` est un service du système qui a été ajouté. il doit être placé dans **`/etc/systemd/system/`**. Il permet la récupération continue du flux d'entrée (et de sortie) du Bluetooth vers un fichier périphérique **`rfcomm0 :`**

```
[Unit]
Description=RFCOMM service
After=bluetooth.service
Requires=bluetooth.service

[Service]
ExecStartPre=/usr/bin/sdptool add --channel=22 SP
ExecStartPre=/bin/chmod 777 /var/run/sdp
ExecStart=/usr/bin/rfcomm watch 0 22
Restart=on-failure
RestartSec=5s
RemainAfterExit=yes

[Install]
WantedBy=multi-user.target
```

La classe active **`CommunicationBluetooth`** effectue une lecture dès que des données ont été reçues :

```
void CommunicationBluetooth::lirePort()
{
    if (!m_pPortSerie->isOpen())
        return;

    QByteArray donnees;

    // lecture des données disponibles
    while (m_pPortSerie->bytesAvailable())
    {
        donnees += m_pPortSerie->readAll();
        this->msleep(20); // en ms
        //::usleep(100000); // cf. timeout
    }

    QString donneesRecues = QString(QString::fromUtf8(donnees.data()));
    if(!donneesRecues.isEmpty())
    {
        #ifndef QT_NO_DEBUG
        qDebug() << Q_FUNC_INFO << QThread::currentThreadId() << this << QString::fromUtf8("Données reçues : ") << donneesRecues;
        #endif
        emit nouvellesDonneesRecues(donneesRecues);
    }
}
```

Ces données sont ensuite envoyées par signaux Qt à **`m_pTrame`** pour être traité et appeler les fonctions correspondantes à la nature des trames.
Sous Qt, le signal doit être **connecté** à un *slot* (une méthode d'une classe). Ici la méthode **`traiterTrame()`** sera exécutée à chaque fois que le signal **`nouvellesDonneesRecues`** sera émis. Ce signal transmet en paramètre les données reçues.

```
connect(m_pCommunicationBluetooth, SIGNAL(nouvellesDonneesRecues(QString)), m_pTrame, SLOT(traiterTrame(QString)));
```

Protocole Bluetooth TTPA

Toute trame commence par l'identifiant **`$TTPA`**. Chacune possède un type sous la forme d'un mot clé.

Le format général d'une trame est le suivant :

`$TTPA:TYPE:{PARAMÈTRE}`

- Les paramètres sont séparés par des '**`:`**'.
- les éléments entre " **`{ }`** " sont ici des paramètres.
- Le délimiteur de fin de trame est **`\r\n`** .

Les différents types de trames pour la communication ECRAN-TTPA et le terminal mobile sont :

`$TTPA:PAUSE`

- Pause la séance.
  `CIhm::pauserSeance()`

`$TTPA:RESUME`

- Reprends la séance.
  `CIhm::reprendreSeance()`

`$TTPA:RESET`

- Réalise un reset de la table (tous les paramètres sauf le nom du joueur).
  `CIhm::resetSeance()`

`$TTPA:QUIT` *(utilisé pour des tests)*

- Quitte le programme.
  `CIhm::quitter()`

`$TTPA:FINSEANCE`

- Signale la fin de la séance.
  `CIhm::finirSeance()`

`$TTPA:START`

- Indique la fin de la phase de configuration avant l'entraînement, bascule sur l'affichage de la table.
  `CIhm::commencerSeance()`

$TTPA:JOUEE *(obsolète, voir **IMPACT**)*

- Informe le programme de l'envoi d'une balle par le robot étant JOUABLE (a rebondi sur la table du coté joueur)
  `CIhm::balleEnJeu()`

$TTPA:IMPACT:{NUM_ZONE}

- Informe le programme d'un impact de balle du côté du robot (opposé au joueur).
  `CIhm::impacterZone(uint8_t zone)`
- Si NUM_ZONE = 0 : le code de **JOUEE** est exécuté.
  `CIhm::balleEnJeu()`

$TTPA:CONNECT:{NOM}

- Paramètre le nom du joueur à afficher et bascule sur l'affichage de la table.
  `CIhm::setInfoConnect(QString nom)`

$TTPA:SETROBOT:{POS_ROBOT} *(sous commande, voir **SETSEANCE**)*

- Définit la position du robot sur la table.
  `CIhm::setZoneRobot(uint8_t zone)`

$TTPA:SETOBJECTIF:{POS_OBJECTIF} *(sous commande, voir **SETSEANCE**)*

- Définit la position de l'objectif sur la table.
  `CIhm::setZoneObjectif(uint8_t zone)`

$TTPA:SETBALLESMAX:{NB_BALLES_MAX} *(sous commande, voir **SETSEANCE**)*

- Définit le nombre de balles pour la séance.
  `CIhm::setBallesMaximum(int balles)`

$TTPA:SETSEANCE:{POS_ROBOT}:{POS_OBJECTIF}:{NB_BALLES_MAX}

- Définit les paramètres de la session en une seule trame, exécute le code de **SETROBOT**, **SETOBJECTIF** et **SETBALLESMAX** dans une même trame.

Exemple de trame:

$TTPA:SETSEANCE:1:2:100

- Ici la position du robot serait en haut à gauche de la table, l'objectif en haut au centre, et le nombre de balles pour la séance serait de 100.

Il s'agit du travail de **`m_pTrame`** de découper et d'appeler les fonctions correspondantes, **`bool`** CTrame**`::traiterTrame(QString)`** est appelée par **`CommunicationBluetooth`** une fois toutes les données de la trame reçues comme dit précédemment.

```
bool CTrame::traiterTrame(QString donneesRecues)
{
    if(donneesRecues.isEmpty())
        return false;

    bool retour = false;

    QStringList listeElements;
    listeElements = donneesRecues.split("$",QString::SkipEmptyParts);

    for(uint8_t i = 0; i < listeElements.length(); i++)
    {
        if (listeElements.at(i).startsWith("TTPA:"))
            retour = gererTrame("$" + listeElements.at(i));
    }
    return retour;
}
```

Ici on découpe les trames reçues à l'aide de la méthode **`split(const QString, int)`** de la classe **`QString`** pour récupérer chaque trame individuellement dans le cas de concaténation de plusieurs trames durant la réception. Chaque trame sera traitée par la suite. L'entête du protocole est vérifié, et les trames sont ensuite gérées dans **`bool CTrame::gererTrame(QString)`**. Elle informe le programme en utilisant les signaux Qt :

```
bool CTrame::gererTrame(QString donneesRecues)
{
    qDebug() << Q_FUNC_INFO << QThread::currentThreadId() << this << QString::fromUtf8("Données reçues : ") << donneesRecues;

    bool trameValide = true;

    if (getTrameLength(donneesRecues) == 2)                 // sans arguments
        trameValide = gererTramesSansParametre(donneesRecues);

    else if (getTrameLength(donneesRecues) == 3)            // 1 arguments
        trameValide = gererTrames1Parametre(donneesRecues);

    //$TTPA:SETSEANCE:[POS_ROBOT]:[POS_OBJECTIF]:[NB_BALLES_MAX]
    else if (getTrameLength(donneesRecues) == 5 && extraireElement(donneesRecues,1).startsWith("SETSEANCE"))
    {
        if ((extraireElement(donneesRecues,2)).toInt() <= 0)    // AUCUN dans le cas d'un negatif ou zero
            emit setZoneRobot(ZONE_AUCUNE);
        else
            emit setZoneRobot((extraireElement(donneesRecues,2)).toInt() - 1);

        if ((extraireElement(donneesRecues,3)).toInt() <= 0)    // AUCUN dans le cas d'un negatif ou zero
            emit setZoneObjectif(ZONE_AUCUNE);
        else
            emit setZoneObjectif((extraireElement(donneesRecues,3)).toInt() - 1);

        emit setBallesMaximum((extraireElement(donneesRecues,4)).toInt());
    }
    else
    {
        qDebug() << Q_FUNC_INFO << "/!\\ TRAME NON RECONNUE /!\\";
        trameValide = false;
    }
    return trameValide;
}
```

L'analyse de trame est réalisé par le nombre de paramètre. Il existe seulement 3 types de trames : celles **sans** paramètre, celles avec **un** seul paramètre et celle (**`SETSEANCE`**) qui possède **3** paramètres.

Ici on vérifie le type de trame seulement, il n'y a pas de paramètre, on envoie par la suite le signal Qt correspondant à la méthode du même nom dans **`CIhm`** permettant l'action demandé par la trame :

```
bool CTrame::gererTramesSansParametre(QString donneesRecues)
{
    bool trameValide = true;
    //$TTPA:JOUEE
    if (extraireElement(donneesRecues,1).startsWith("JOUEE"))
    {
        emit balleEnJeu();
    }
    //$TTPA:START
    else if (extraireElement(donneesRecues,1).startsWith("START"))
    {
        emit commencerSeance();
    }
    //$TTPA:FINSEANCE
    else if (extraireElement(donneesRecues,1).startsWith("FINSEANCE"))
    {
        emit finirSeance();
    }


        [...]
}
```

Dans cette méthode, le type de trame est vérifié ainsi que son paramètre :

```
bool CTrame::gererTrames1Parametre(QString donneesRecues)
{
    bool trameValide = true;

    // $TTPA:CONNECT:[nom]
    if (extraireElement(donneesRecues,1).startsWith("CONNECT"))
    {
        emit setInfoConnect(extraireElement(donneesRecues,2));
    }
    //$TTPA:SETROBOT:[POS_ROBOT]
    else if (extraireElement(donneesRecues,1).startsWith("SETROBOT"))
    {
        if ((extraireElement(donneesRecues,2)).toInt() <= 0)    // Adaptation au systeme de la table
            emit setZoneRobot(ZONE_AUCUNE);
        else
            emit setZoneRobot((extraireElement(donneesRecues,2)).toInt() - 1);
    }

         [...]
}
```

`QString extraireElement(QString, const int)` réalise l'extraction de l'élément demandé (paramètre), le premier paramètre est la trame entière, le second est la position de l'élément voulu :

```
QString CTrame::extraireElement(QString donneesRecues, const int iElement)
{
    if(donneesRecues.isEmpty())
        return QString();

    QStringList listeElements;
    listeElements = donneesRecues.split(":");

    return listeElements.at(iElement).trimmed();
}
```

`trimmed()` est une méthode de **`QString`** permettant la suppression de caractères d'espacements tél **`'\n'`** dans notre cas **`"\n\r"`** en fin de trame.

L’Interface graphique

Le programme est composé de plusieurs fenêtres créé à l’aide d’un **`QStackedWidget`**

L’affichage “LOGO” est la première page, il s’agit de l'écran de connexion, elle est affichée au lancement du programme et à la déconnexion du terminal mobile.

Dans cet état, les trames attendues sont:

- **`$TTPA:CONNECT:{NOM}`**
- **`$TTPA:SETSEANCE:{POS_ROBOT}:{POS_OBJECTIF}:{NB_BALLES_MAX}`**
- **`$TTPA:START`**

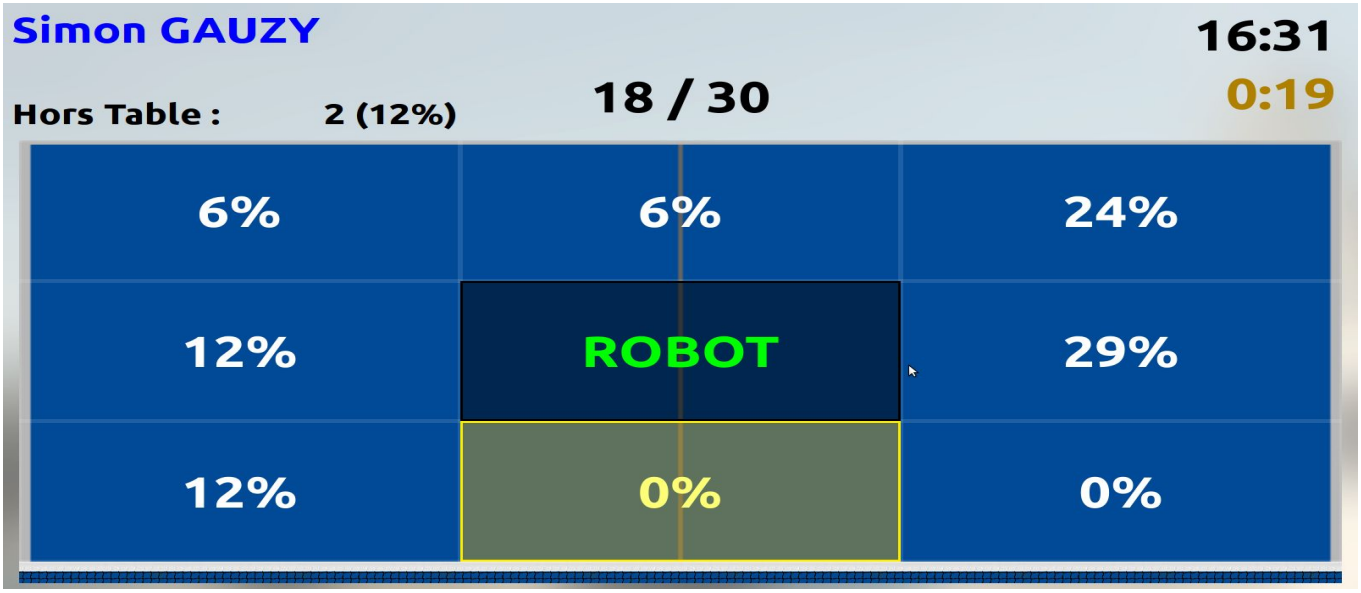
Suite à la connexion du terminal mobile et à l'envoi de la trame ci-dessous, on affiche le nom du joueur ainsi que le texte d'attente de configuration :

- **$TTPA:CONNECT:Simon GAUZY**

Enfin la trame de configuration puis la trame de démarrage de séance :

- **$TTPA:SETSEANCE:5:8:30**
- **$TTPA:START**

Suivi du déroulement de la séance configurée pour le joueur :

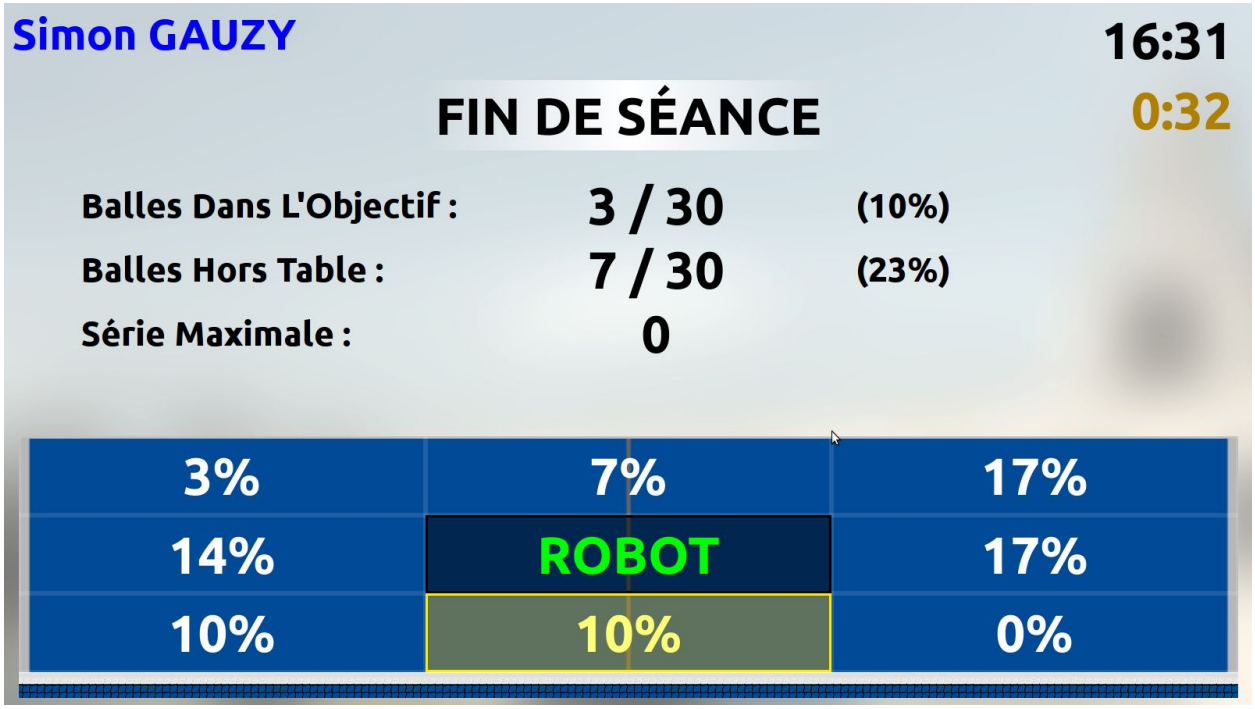


Cet affichage “TABLE” est la page affichée une fois le terminal mobile connecté et la séance paramétrée, il contient :

- L’affichage des balles jouées (en haut au centre)
- La répartition des balles dans les 9 zones de la table mais aussi les impacts en temps réel sur cette même table
- La “Zone d’objectif” (en jaune, sur la table)
- La “Zone du robot” (en vert sur noir, sur la table)
- Les balles hors table (en haut à gauche de la table)
- Le nom du joueur (en haut à gauche)
- L’Heure et le temps de la séance écoulée (en haut à droite)

Les trames pouvant être utilisées dans cet écran par le Mobile-TTPA sont :

- **`$TTPA:IMPACT:{NUM_ZONE}`**
- **`$TTPA:RESET`**
- **`$TTPA:FINSEANCE`**



l'affichage "RECAP" (tableau récapitulatif) est ensuite affiché, il résume la séance, montrant des statistiques plus détaillées.

Ici il est possible d'utiliser **`$TTPA:START`** pour redémarrer une séance avec la même configuration, ou encore de configurer cette prochaine séance avec :

`$TTPA:SETSEANCE:{POS_ROBOT}:{POS_OBJECTIF}:{NB_BALLES_MAX}`

ou se déconnecter du système.

Suite à la déconnexion, la table est réinitialisée (*reset*) et la page de connexion est de nouveau affichée.

Affichage de la table et calculs de la séance

La classe **`CTable`** génère une table composé de 9 **`QLabel`** (Widget Qt permettant de créer des zones de texte) ainsi que 2 autres purement graphique pour le filet et la texture de la table. L'objet **`m_pTable`** est instancié dans le constructeur de **`CIhm`** et son objet parent est l'objet ihm (this). Il est ensuite placé dans le layout **`m_pHBLayoutTable`**. Un *layout* est un conteneur Qt qui fournit un système de positionnement de *widgets*. Ici le layout permet un positionnement **h**orizontal sous forme de **b**oîte (xxxHBxxx) :

```
m_pTable = new CTable(this);
m_pHBLayoutTable->addWidget(m_pTable);
```

`addWidget()` permet de déplacer un Widget, il s'agit d'une méthode présente dans les conteneurs Qt.

Il est utilisé également par la suite au moment d'afficher l'écran récapitulatif de la séance afin de ne pas instancier une nouvelle table pour l'occasion, comme dit précédemment le widget est "déplacé", il n'est pas instancié à nouveau.

Les zones sont instanciées dans le constructeur de **`CTable`** puis sont placées dans un **`QVector`** (équivalent dans Qt du **`Vector`** avec plus de fonctionnalitées).

```
QLabel* pZone;
for(uint8_t i=0; i < NB_ZONES; i++)
{
    pZone = new QLabel(this);
    pZone->setText("Zone " + QString::number(i+1));
    pZone->setFont(m_font);
    pZone->setStyleSheet(CSS_FOND_INACTIF);
    pZone->setAlignment(Qt::AlignCenter);

    //-----------------------------------
    m_pZones.push_back(pZone);
    m_pGridLayout->addWidget(pZone, (i/3), (i%3));
}
```

En même temps, la fonte d'écriture est appliqué ainsi que le StyleSheet CSS pour les couleurs et autre paramètres graphiques.

Le texte de ces zones sont ensuite changé en fonction de la séance, ainsi que leur StyleSheet pour afficher la zone touchée / placement du robot / zone objectif.

Les StyleSheets sont définies en tant que constantes (**`#define`**) :

```
#define CSS_FOND_INACTIF    QString::fromUtf8("QLabel\n{\nbackground-color: rgba(50, 150, 255, 0);\nborder: 3px solid rgba(255,255,255,30);\ncolor: #FFFFFF;\n}")
#define CSS_FOND_ACTIF      QString::fromUtf8("QLabel\n{\nbackground-color: rgb(0, 150, 50);\nborder: 3px solid #00FF00;\ncolor: #FFFFFF;\n}")
#define CSS_FOND_RATE       QString::fromUtf8("QLabel\n{\nbackground-color: rgb(175, 50, 25);\nborder: 3px solid #FF0000;\ncolor: #FFFFFF;\n}")

#define CSS_FOND_ROBOT      QString::fromUtf8("QLabel\n{\nbackground-color: rgba(0, 0, 0, 120);\nborder: 3px solid #000000;\ncolor: #00FF00;\n}")
#define CSS_FOND_OBJECTIF   QString::fromUtf8("QLabel\n{\nbackground-color: rgba(200, 160, 30, 120);\nborder: 3px solid #FFEE00;\ncolor: #FFFF77;\n}")
```

Il s'agit de la classe **`CTrame`** qui envoie un signal à **`CIhm`** qui par la suite appelle la méthode **`bool CTable::impacterZone(uint8_t numeroZone)`** :

```
bool CTable::impacterZone(uint8_t numeroZone)
{
    qDebug() << Q_FUNC_INFO;
    if (numeroZone >= NB_ZONES || numeroZone == m_iZoneRobot)
        return false;

    if (m_iBallesBonnes + m_iBallesHorsTable >= m_iBallesTotal)
        return false;

    m_bBalleCoteTablePrec = m_bBalleCoteTable;
    m_bBalleCoteTable = true;

    qDebug() << Q_FUNC_INFO << numeroZone;
    m_iZoneToucheePrec = m_iZoneTouchee;
    m_iZoneTouchee = numeroZone;

    m_iBallesBonnes++;
    m_iBallesDansZone[numeroZone]++;

    rafraichirNbBallesZone();

    if (numeroZone != ZONE_AUCUNE)
    {
        if (numeroZone == m_iZoneObjectif || m_iZoneObjectif == ZONE_AUCUNE)
        {
            m_pZones[numeroZone]->setStyleSheet(CSS_FOND_ACTIF);
            m_iBallesEnchainees++;
        }
        else
        {
            m_pZones[numeroZone]->setStyleSheet(m_fondRate);
            m_iBallesEnchainees = 0;
        }

        m_pZones[numeroZone]->setFont(m_fontBig);
    }
    else
        m_iBallesEnchainees = 0;

    if (m_iBallesEnchainees > m_iBallesEnchaineesMax)
        m_iBallesEnchaineesMax = m_iBallesEnchainees;

    m_iZoneTouchee = numeroZone;
    QTimer::singleShot(DELAI_COUP, this, SLOT(rafraichirInactif()));

    return true;
}
```

Des vérification sont réalisé afin de ne pas permettre de balles impossibles (ex: Robot placé en 2, une balles impacte cette même zone, c'est impossible).

On récupère également le côté que la dernière balle a impacté afin de savoir si elle n'a jamais été renvoyé

Le nombre de balles mise en jeu par le robot, ainsi que le nombre de balles dans chaque Zone

Ainsi que l'affichage des zones touchées, passage a une Zone en vert quand l'objectif est touché, en rouge quand il ne l'est pas

Et l"incrémentation du nombre d'enchainement réussit successivement.

Diagrammes

Diagramme de déploiement

Le système ECRAN-TTPA nécessite d'être raccordé à une Télévision en HDMI ainsi que de communiquer en Bluetooth à l'aide d'un port Série rfcomm virtuel

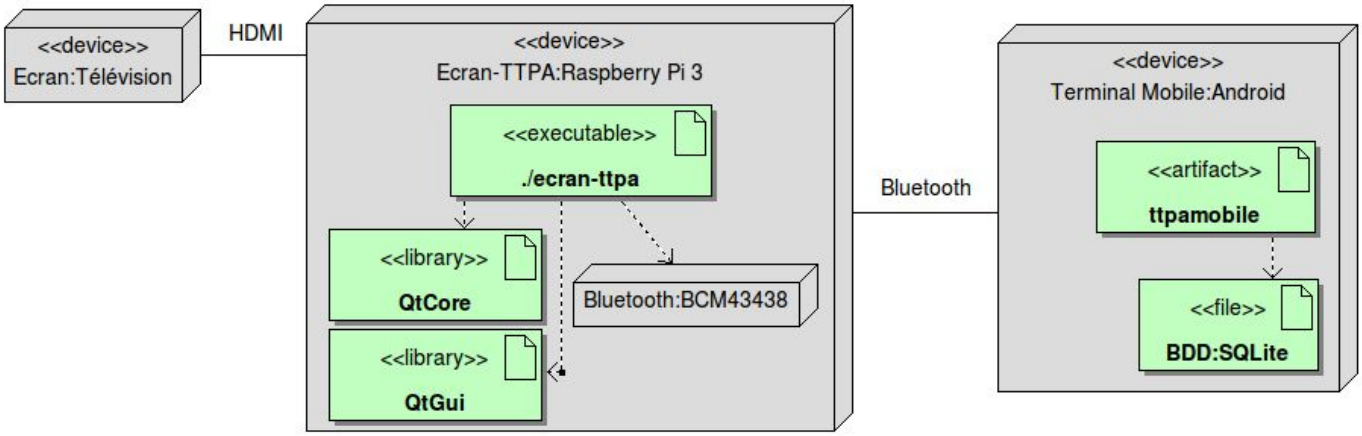
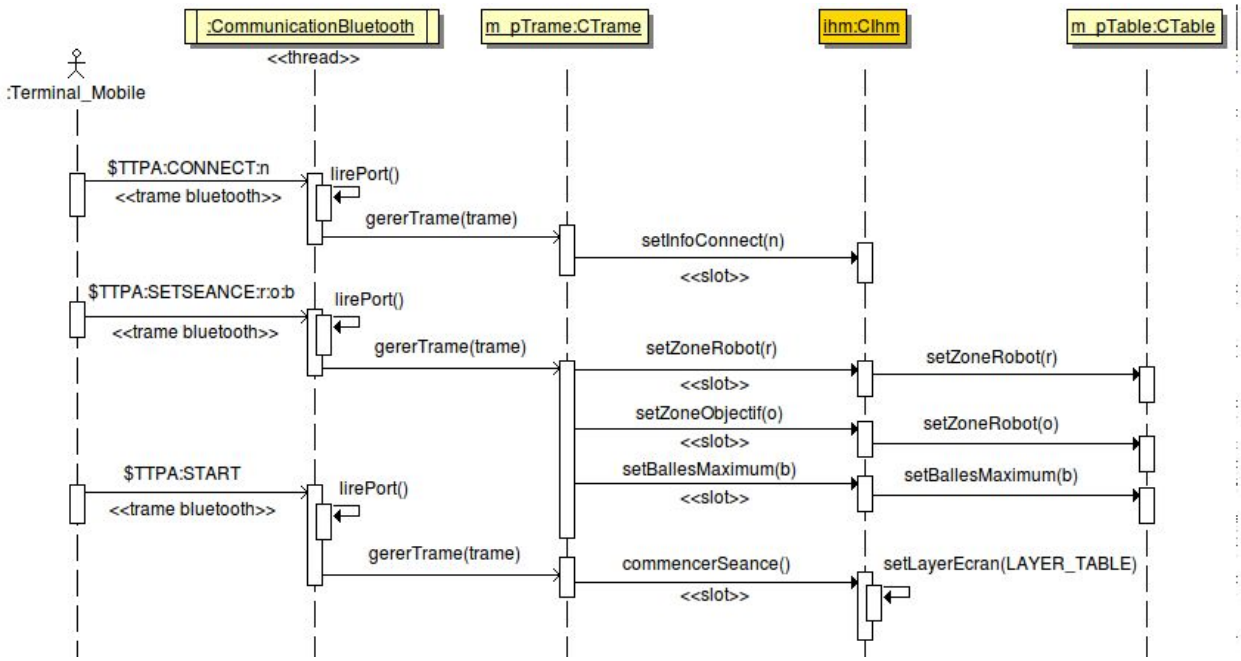


Diagramme de Séquence : Démarrage et Configuration

Pour chaque trame reçue, la méthode **`gererTrame()`** est appelée. Elle vérifie si la trame commence par l'identifiant **`$TTPA`**. Ensuite, elle extrait le type de trame (`CONNECT`, `SETSEANCE`, `START`).

La configuration est envoyée par le terminal Android.

`$TTPA:CONNECT:{NOM}` est alors envoyé à la Raspberry pour l'identification du joueur, puis **`$TTPA:SETSEANCE:{POSITION_ROBOT}:{POSITION_OBJECTIF}:{NOMBRE_BALLES}`** pour la configuration de la séance, puis enfin, **`$TTPA:START`** pour la démarrer.



La configuration est réalisée sur le terminal mobile par l'envoi de trames au programme, la classe **`m_pCommunicationBluetooth`** reçoit ces trames depuis le port **`rfcomm`** et envoie par signal les trames à **`m_pTrame`** afin de les découper pour récupérer leurs paramètres et par la suite les envoyer à l'IHM (par signaux Qt) pour effectuer les fonctions correspondantes à ces trames.

(voir page 14 pour les informations relatives à ces trames)

Diagramme de Séquence : Déroulement d'une Séance

Suite à l'envoi de balles par le robot, ses balles impactent le côté du joueur, envoyant **`$TTPA:IMPACT:0`** puis par le renvoi de celles-çi par le joueur, **`$TTPA:IMPACT:{NUM_ZONE}`** comptabilisant les zones touchées. l'appel de **`$TTPA:FINSEANCE`** termine la séance et affiche la page récapitulative de performance du joueur lors de la séance

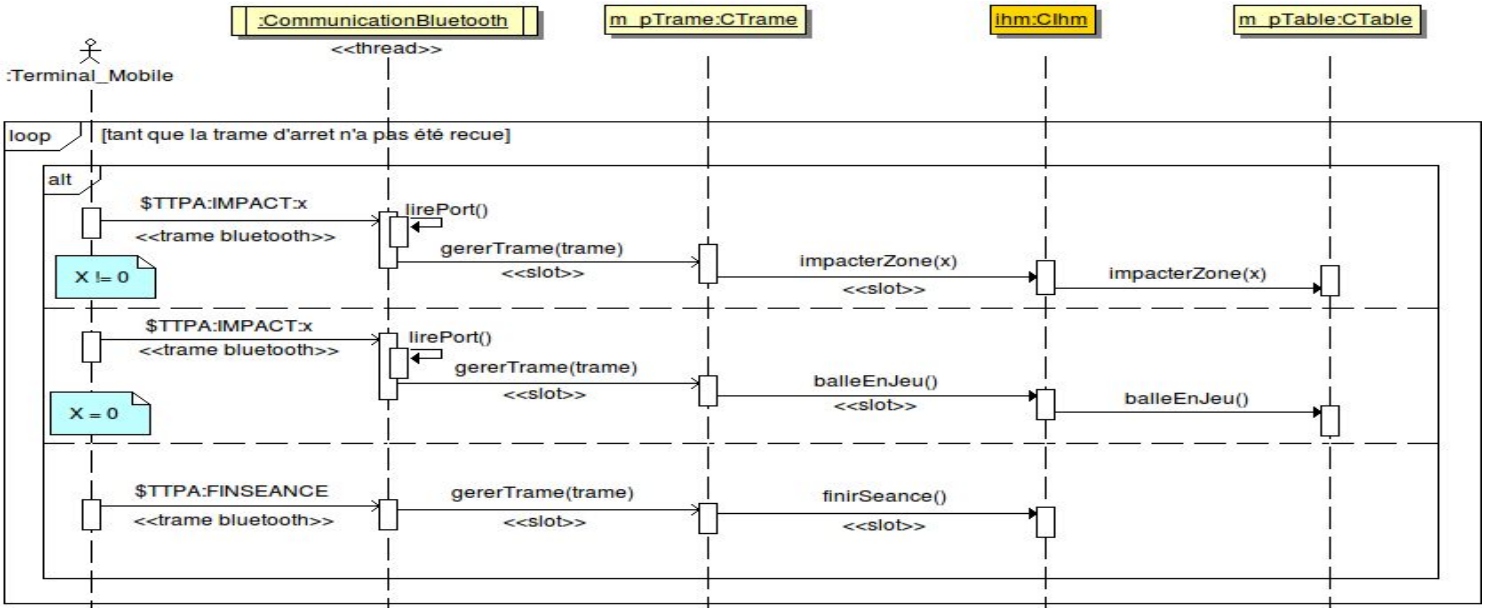
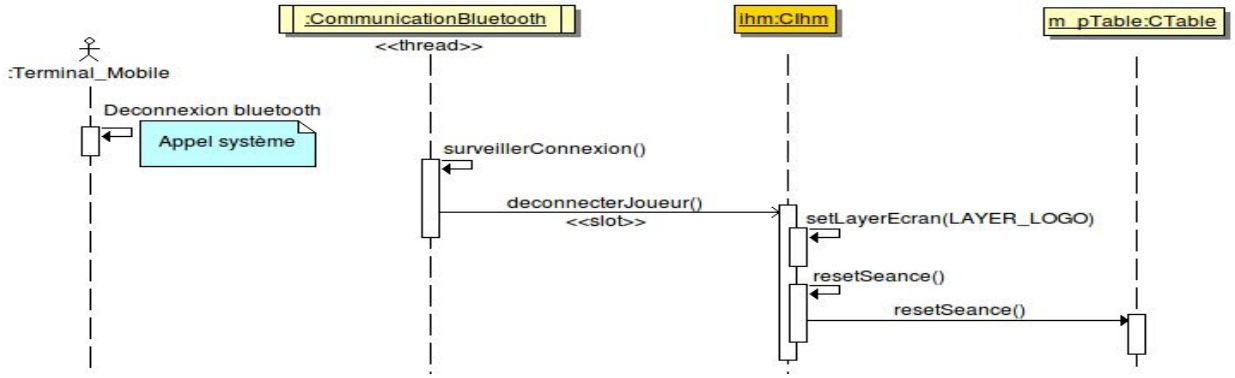


Diagramme de Séquence : Déconnexion du bluetooth

Il s'agit de la déconnexion du bluetooth depuis le terminal android de façon brusque, celle-ci entraîne l'écran-TTPA à reset la séance et d'afficher l'écran d'attente de connexion.



Dans ces diagrammes, les possibilitées de déroulement d'une séance sont tous mis en oeuvre.

Tests de validation

Description	OUI	NON
Le système d'exploitation est installé et fonctionnel	**x**	
L'écran est configuré en mode "kiosque"	**x**	
La zone d'impact est identifiée et affichée en temps réel	**x**	
Les données de la séance (le pourcentage de balles par zones et nombre de pourcentages de balles bonnes) sont affichées en temps réel	**x**	
Les liaisons sans fil sont opérationnelles	**x**	
Les informations sont affichées en fin de séquence	**x**	

Annexes

Options de Lancement

Des options de lancement sont disponibles pour des facilitées de développement :

-windowed	Lance le programme en mode fenêtré. (peut être combiné)
-console	Ajoute une console permettant d'envoyer des trames manuellement.
-dev	Mode de test sans terminal android. (OBSOLÈTE, voir -console)
-demo	Mode démonstration de la table, "-norobot" pour désactiver le robot