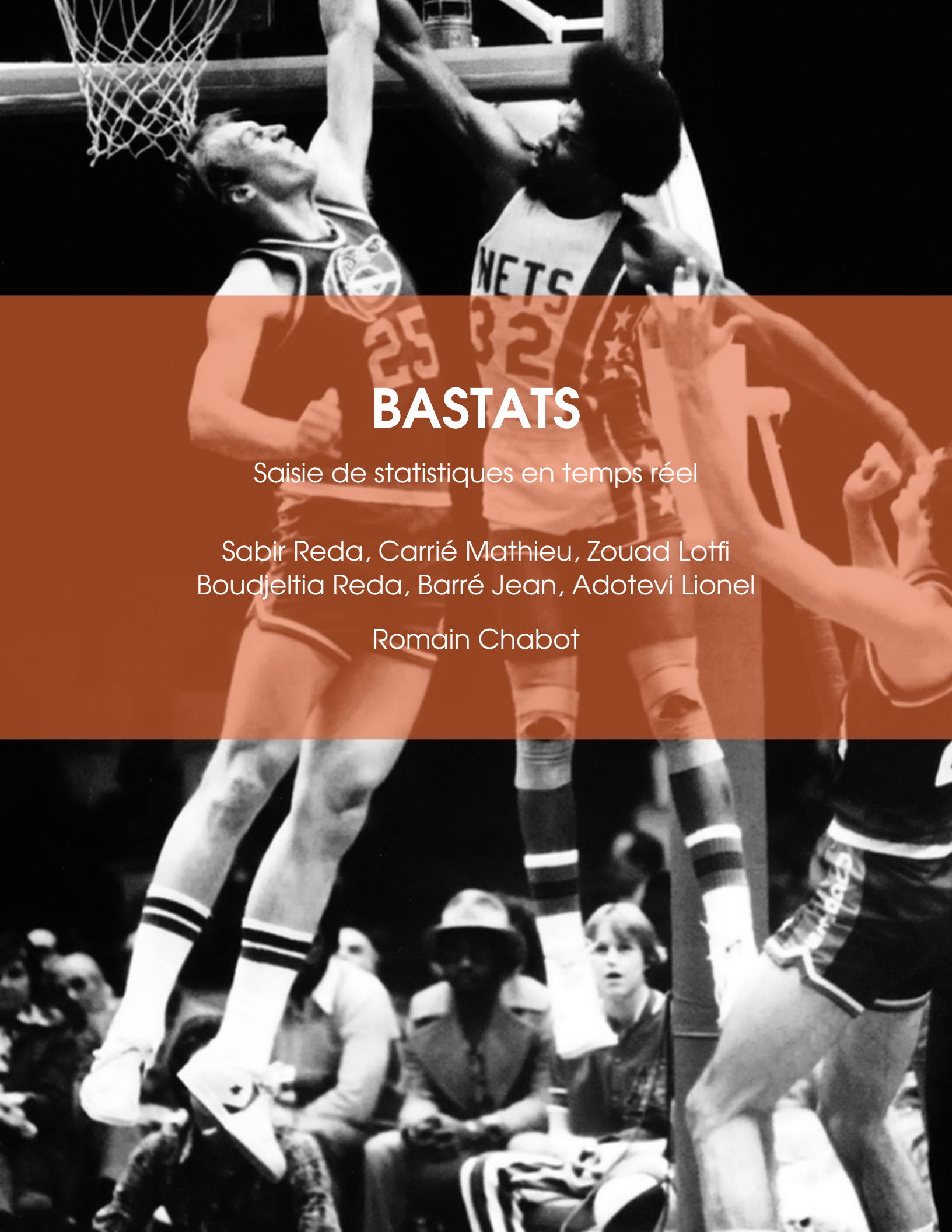


BASTATS

Saisie de statistiques en temps réel

Sabir Reda, Carrié Mathieu, Zouad Lotfi
Boudjeltia Reda, Barré Jean, Adotevi Lionel

Romain Chabot

PROJET DE FIN DE DEUXIÈME ANNÉE, ENSEIRB-MATMECA

Projet réalisé sous la supervision de M. Yves Métivier dans le cadre d'un Master en Informatique à l'ENSEIRB-MATMECA, Bordeaux (France).

30 Mars 2015

BASTATS

Table des matières

1 Introduction 5

2 Expression des Besoins 7

2.1 Motivation 7

2.2 Étude de l'existant 7

2.3 Besoins fonctionnels 8

2.3.1 Définitions 8

2.3.2 Besoins et contraintes 8

3 Stratégies de conception 11

3.1 Spécificités du développement Android 11

3.1.1 Comment fonctionne une application Android ? 11

3.1.2 La création des vues utilisateurs 12

3.1.3 L'utilisation des fragments 13

3.1.4 A propos de l'environnement de développement 14

3.2 Modélisation de la base de données 14

3.3 Format d'import et d'export des données 17

3.3.1 Entre le site et l'application 17

3.3.2 La feuille de match 18

3.4 Conception de l'interface 18

3.4.1 Méthode de fonctionnement 18

3.4.2 Interface de match 19

3.4.3 Lancement d'un match 20

3.4.4 Interface de navigation d'un tournoi 21

4 Réalisation de l'interface 23

4.1 Lancement d'un match 23
4.1.1 Sélection des équipes 24
4.1.2 Possibilités d'ajout de nouveaux joueurs 24
4.1.3 Sélection des joueurs pour un match 24

4.2 Navigation dans un tournoi 25
4.2.1 Accueil Tournoi 25
4.2.2 Nouveau Tournoi 25
4.2.3 Sélection des équipes pour le tournoi 25
4.2.4 Menu de navigation 26
4.2.5 Création d'une phase 26
4.2.6 Navigation poule 26
4.2.7 Navigation tableau 27

4.3 Interface de match 27
4.3.1 Module 1 : Barre supérieure 28
4.3.2 Modules latéraux : Joueurs et saisie des statistiques 28
4.3.3 Module central 28
4.3.4 Améliorations de l'interface 29

4.4 Visualisation des statistiques 29

4.5 Historique Final 29

5 Évaluation du produit final 31

5.1 Comparaison avec le cahier des charges 31
5.1.1 Interface de match 31
5.1.2 Statistiques 31
5.1.3 Import/Export 31
5.1.4 Gestion de tournoi 32

6 Conclusion 33

A Document des spécifications 35

B Manuel d'utilisation 47

C Dessins papiers 83

D Schéma de la base de données 89

BASTATS

1. Introduction

Ce PFA avait pour objectif de développer une application destinée à une utilisation sur tablette Android et permettant de saisir des statistiques sur les joueurs lors d'un match de Basketball. La personne chargée de la saisie des statistiques se tient sur le bord du terrain munie de la tablette et doit pouvoir enregistrer les actions effectuées par les joueurs des deux équipes en temps réel pendant le match. Cela pose de nombreuses contraintes, notamment sur la vitesse d'exécution des actions sur la tablette. Ce point sera exploré plus en détails dans la suite de ce document.
Nous avons choisi de baptiser notre application "*Bastats*", simple contraction de Basket et Stats qui permet de référer l'application de manière plus rapide.

Ce rapport sera décomposé en plusieurs parties. Dans un premier temps, nous exposerons les besoins établis pour developper notre application en nous basant sur une étude d'applications similaires et les échanges avec M.Ilcinkas. Ensuite, il nous faudra poser le contexte en définissant brièvement le fonctionnement d'une application Android afin d'expliquer dans les parties suivantes le fonctionnement de la base de données et la méthode de conception de l'interface.

Après avoir expliqué notre processus de travail, nous proposerons une évaluation aussi objective que possible de l'application en se référant au cahier des charges initial, et en analysant les déviations observées.

Enfin, le document des spécifications et un manuel d'utilisation de l'application seront disponibles en annexe, ainsi que des esquisses papier de l'interface utilisateur qui permettront d'illustrer des points évoqués dans l'ensemble du rapport.

2. Expression des Besoins

2.1 Motivation

Lors d'un match de basketball, il peut être important pour un entraîneur d'évaluer les performances des joueurs de l'équipe afin de pouvoir faire des ajustements si besoin sur l'effectif de jeu. Une des manières les plus simples et évidentes d'évaluer les performances est le suivi et l'enregistrement de différentes statistiques pour chaque joueur. L'apparition des tablettes depuis quelques années peut faciliter cette tâche de saisie au bord du terrain. Moins encombrantes qu'un ordinateur portable et plus commodes que le bloc notes classique pour saisir une quantité relativement importante de données, une application tablette parait alors être un bon moyen d'effectuer cette tache.

De plus, lors d'un match professionnel de basketball, des personnes sont affectées uniquement à la saisie de statistiques. Ce type d'organisation n'existe pas forcément dans des matchs de niveau amateur et une application mobile pourrait permettre à n'importe qui de saisir les statistiques, la seule contrainte étant de posséder une tablette et de connaître les règles du jeu. Aussi, si l'application permet une saisie rapide des statistiques de jeu, cela peut fournir l'avantage de ne nécessiter qu'une seule personne pour saisir les statistiques des 2 équipes, ce qui est intéressant notamment pour des structures amateur. Une saisie rapide peut également éviter la perte de données si la personne saisissant les statistiques manque de temps pour entrer un enchaînement des actions se déroulant rapidement.

2.2 Étude de l'existant

De nombreuses applications permettant de saisir des statistiques existent déjà et proposent des fonctionnalités intéressantes :

- Possibilité de saisir pour chaque joueur les différentes actions qu'il peut avoir effectué pendant le match (Tir à 2 points/3 points, Interceptions, Rebonds, etc...).
- Pour les tirs d'un joueur, avoir la possibilité d'indiquer de quel endroit du terrain le tir a été effectué.

— Possibilité de gérer les remplacements pendant le match.
— Possibilité de saisir les statistiques pour les 2 équipes participant au match.

Si de nombreux éléments peuvent être inspirés des exemples d'applications existantes, celles-ci possèdent aussi des défauts pour être utilisées à un niveau amateur. En effet, les remarques suivantes ont été faites sur les applications existantes :

— Saisir une action pour un joueur nécessite plusieurs clics, ce qui peut être handicapant lors de phases de jeu rapides.
— Lorsqu'il est possible de saisir les actions des joueurs pour les 2 équipes du match, un changement d'équipe est requis, ce qui constitue une perte de temps non négligeable car nécessitant un voir plusieurs clics.
— Lorsque la gestion des remplacements est possible, seuls les 5 joueurs de champs sont affichés. A un niveau amateur, cela peut être gênant puisque les changements ne sont font pas toujours de manière officielle et un joueur peut donc être présent sur le terrain sans que le statisticien ne le remarque.

Pour construire notre application, nous nous sommes donc inspirés des différentes applications existantes, que cela soit des applications tablettes ou bien des interfaces destinées à un usage sur ordinateur comme des sites web. Par exemple, l'interface de saisie des statistiques est une adaptation de ce qui est disponible sur le site `lmb.monocycle.info` avec lequel notre application doit s'interfacer au niveau de l'import/export des données.

2.3 Besoins fonctionnels

2.3.1 Définitions

Avant d'exposer la liste des besoins, il est nécessaire de poser quelques définitions afin d'éviter tout malentendu ou zone de flou sur ce qui est écrit dans le rapport.

— **Rapidité :** On quantifie la rapidité d'une saisie en comptant le nombre de clics effectués par l'utilisateur :
  1 clic = instantané
  2 clics = rapide
  3 clics et plus = lent
— **Equipe offensive :** Equipe en possession du ballon.
— **Equipe défensive :** Equipe qui n'a pas le ballon.
— **Turnover** : Perte de balle attribuée à un attaquant.
— **Steal** : Vol de balle ou interception d'un défenseur. A noter qu'un `steal` d'un défenseur implique un `turnover` chez un attaquant. Mais qu'un `turnover` peut avoir lieu sans qu'il y ai eu de `steal`.
— **Rebond** : Récupération du ballon par un joueur d'une des deux équipes après une tentative de panier ayant échouée.
— **Assist** : Passe décisive donnée à un coéquipier.
— **Block** : Contre effectué par un défenseur sur un tir d'un attaquant.

2.3.2 Besoins et contraintes

Suite aux différentes observations sur l'existant et aux exigences de M.Ilcinkas, nous avons établit une liste de besoins à laquelle notre application devra répondre.

Ecran de saisie des statistiques

L'anatomie de cet écran sera exposée en détail plus loin de le document. Pour le moment, on se référera à cet écran par l'intermédiaire de la figure 3.8.

— Certaines actions doivent être accessible en accès instantané comme par exemple les 2 points ou les rebonds.
— Les actions qui ne sont pas en accès instantané doivent être disponibles en "accès rapide" : l'enregistrement de l'action s'effectue en 2 clics, on sélectionne l'action puis le joueur.
— Les actions peuvent être séparées en actions offensives et défensives. Les actions offensives ne pourront être attribuées qu'à l'équipe offensive (idem pour la défense).
— Ainsi, les actions changeant la possession de la balle devront changer automatiquement la possession (Tirs réussis, Rebonds défensifs et Interceptions par exemple). La possession devrait aussi pouvoir être changée manuellement.
— Chaque action doit être ajoutée dans l'historique et doit pouvoir être annulée en cas d'erreur de saisie.
— Chaque action doit être datée dans le match afin de pouvoir avoir un historique dans l'ordre chronologiques des actions effectuées.

Gestion de tournoi

L'application est destinée à être utilisée pour saisir des statistiques lors de matches de basketball. Or, il arrive souvent que ces matches aient lieu dans le cadre d'un tournoi ou d'un championnat. Ainsi, plusieurs fonctions peuvent être ajoutées à l'application pour prendre en compte ce genre d'événements :

— Un tournoi doit pouvoir être crée à partir de l'application. L'application doit également permettre de créer des poules et d'y ajouter des équipes, puis de lancer les matches générés pour le tournoi.
— Un championnat doit pouvoir être crée depuis la l'application. De la même façon que pour le tournoi, l'application doit permettre de sélectionner les différentes équipes participant au championnat ainsi que de générer les différents matches du championnat.
— A chaque lancement de match, on veut pouvoir sélectionner les 2 équipes participant au match et dans chaque équipe, pouvoir sélectionner les joueurs participant effectivement au match.

Visualisation des statistiques

— Un écran permettant la visualisation des statistiques doit pouvoir être accessible à tout moment d'un match ainsi qu'une fois le match terminé. Cet écran doit afficher pour chaque équipe, les statistiques de chaque joueur pour le match donné. L'anatomie de cet écran et les informations qui y sont présentées sont basées sur des feuilles de statistiques classiques présentent sur de différents sites spécialisés tels que `stats.nba.com`.
— Un export des ces statistiques dans des fichiers externes doit également être possible (par exemple au format PDF) afin de faciliter l'analyse et le partage des statistiques.

Import/Export

L'application doit pouvoir s'interfacer avec le site `lmb.monocycle.info`. Même si il peut être possible de créer un tournoi de toutes pièces sur la tablette, il est plus aisé et plus rapide d'effectuer cette opération sur un ordinateur. En effet, l'application sur tablette est vue comme un outil portable de saisie des données pour les réintégrer dans un système plus puissant, tel que le site `lmb.monocycle.info` par exemple. Les contraintes sur l'import et l'export sont les suivantes :

— L'import de données depuis le serveur doit être possible. Par données on entend un ensemble d'équipes et donc de joueurs.
— L'export de données vers un fichier externe doit être possible également. Ici, on peut distinguer 3 types d'export. Le premier se charge d'exporter seulement le résultat d'un match. Le deuxième se charge d'exporter le résultat d'un match ainsi que les statistiques des joueurs lors de ce match dans un fichier destiné à être reçu par le serveur (dans notre cas un fichier JSON). Le dernier type d'export génère un fichier PDF où les statistiques des joueurs de chaque équipes pour un match donné sont présentées dans un tableau destiné à être facilement lisible et exploitable.

3. Stratégies de conception

3.1 Spécificités du développement Android

La grande particularité de ce projet est donc de développer une application pour tablette Android. Cela nous a contraint à apprendre rapidement les spécificités du développement Android, sachant qu'aucun de nous n'avait pas d'expérience dans ce domaine. Android utilise le langage de programmation Java que nous connaissons, ce qui a facilité notre apprentissage. Dans cette partie nous allons présenter les principaux concepts imposés par le développement Android et certaines de ses spécificités.

3.1.1 Comment fonctionne une application Android ?

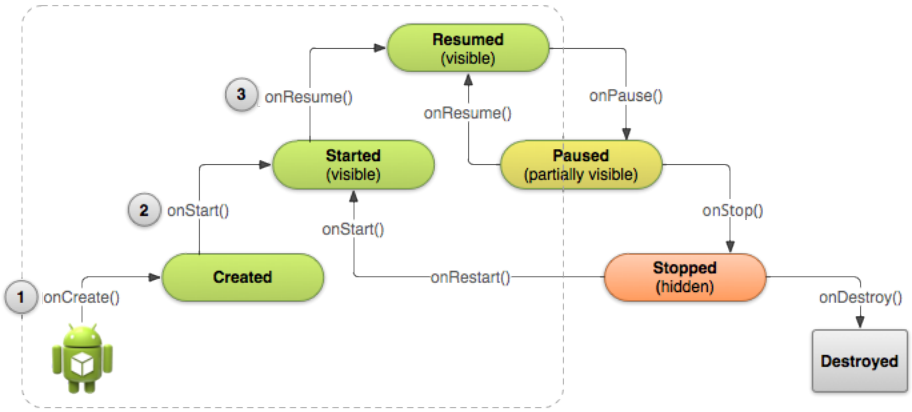


FIGURE 3.1 – Cycle de vie d'une activité

Le fonctionnement des applications Android est géré par les **activités**. Les activités sont la composante principale des applications Android et peuvent être assimiler à un programme qui s'exécute. Une activité permet de gérer la vue afficher à l'écran et les comportements du programme

en fonction des actions effectuées par l'utilisateur sur l'écran (ex : clic sur l'écran). Chaque activité a un cycle de vie qui lui est propre.

Comme l'on peut le voir sur la figure 3.1, une activité possède 7 méthodes indispensables afin de gérer son cycle de vie. La plupart du temps seule la méthode `onCreate()` est implémentée, car c'est là que la vue est générée et que l'on peut définir les comportements à l'exécution. Les 6 autres méthodes possèdent toutes un comportement par défaut, mais il est possible de redéfinir chacune selon les besoins.

Bien comprendre le cycle d'une activité est important dans la mesure où plusieurs activités vont cohabiter au sein d'une seule application. L'application gère les activités de manière cachée à l'aide d'un pile appelée `back stack`.

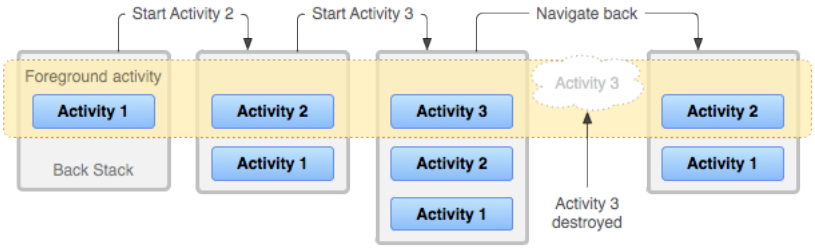


FIGURE 3.2 – Illustration de la back stack

Les activités sont empilées les unes sur les autres au fur et à mesure que l'utilisateur déclenche des actions sur l'écran. Quand une nouvelle activité est démarrée, l'activité précédente appelle sa méthode `onPause()` puis s'arrête en appelant `onStop()` pour laisser place à la nouvelle activité qui va afficher un nouvel écran. Ensuite lorsque le bouton `retour` est utilisé, l'activité sur le haut de la pile est définitivement détruite avec `onDestroy()` et l'on revient à l'activité précédente qui va déclencher ses méthodes `onRestart()`, `onStart()` puis `onResume()` pour reprendre au même endroit où elle s'était arrêtée.

Cette gestion des activités via une pile nous incite à développer l'application en prêtant attention à ne pas créer trop d'activités simultanément sur la tablette afin d'économiser les ressources disponibles, qui sont plus limités sur une tablette ou un smartphone.

3.1.2 La création des vues utilisateurs

Comme nous l'avons expliquer précédemment, l'activité permet de gérer l'affichage de la vue sur l'écran de la tablette. Pour générer cette affichage, l'activité procède à une opération appelée `désérialisation`. Pour faire simple cela consiste à transformer un objet décrit en langage XML en un objet Java. Cette opération s'effectue dans la méthode `onCreate()` de l'activité. Ainsi les différents composants d'une vue sont instanciés à partir d'un fichier XML appelé `layout`. Ce fichier est généré à l'aide d'une interface graphique d'Android Studio qui aide à placer grossièrement les composants sur la vue, mais aussi à visualiser une vue à partir de code XML que nous avons pu écrire. Normalement une activité est liée à une seule et unique vue. Ainsi pour chaque nouvelle écran de l'application, nous devons créer une nouvelle activité, tout en supprimant l'activité précédente si celle-ci devient inutile par la suite. Le fait qu'une activité ne puisse permettre d'afficher qu'une seule vue est contraignant au niveau du nombre de fichiers crées. Pour limiter cet effet il est possible d'utiliser des fragments qui vont permettre de partitionner les vues d'une activité.

3.1.3 L'utilisation des fragments

Les fragments ont été introduit en Android pour permettre à une activité d'adapter l'affichage selon que l'application soit destinée à une tablette ou un téléphone comme illustré sur le schéma suivant :

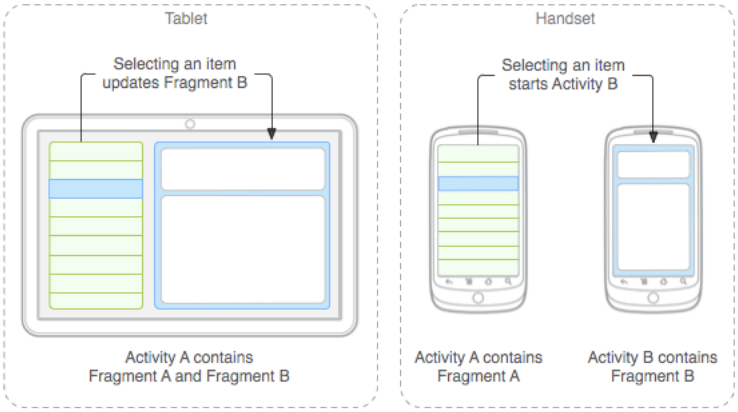


FIGURE 3.3 – Objectif des fragments

Les fragments permettent de cette façon de diviser une activité en plusieurs composants qui seront encapsulés. En effet un fragment représente un objet qui possède ses propres méthodes. Cela permet à l'activité de ne pas avoir à gérer les fonctions d'affichage par exemple. Ainsi un fragment peut être réutilisable dans plusieurs activités différentes et permet d'éviter de copier coller le même code d'une activité à l'autre. La problématique posée est alors de faire communiquer le fragment avec l'activité qui le contient. Pour cela, le fragment possède une interface avec des méthodes de callback. Ces méthodes de callback sont appelées dans le fragment. L'activité doit alors implémenter les fonctions de l'interface pour définir quel comportement adopter.

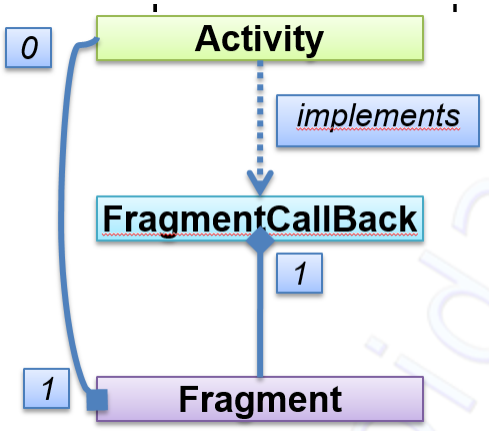


FIGURE 3.4 – Interaction entre le fragment et l'activité

3.1.4 A propos de l'environnement de développement

Avant le démarrage du PFA, les applications Android étaient développées à l'aide de l'IDE Eclipse pour la plupart. Mais cela n'était pas très pratique pour pour plusieurs raisons. En décembre 2014, Google a sorti la version stable de son IDE Android Studio. Android Studio a été construit spécifiquement pour répondre aux besoins des développeurs d'applications Android (ce qui n'est pas le cas d'Eclipse). Nous avons donc choisi de développer notre projet dans Android Studio.

Android Studio permet notamment de générer automatiquement le squelette d'un projet Android. L'IDE permet aussi une compilation automatique et de lancer aisément le programme en appuyant sur un bouton RUN.

3.2 Modélisation de la base de données

Tout d'abord, dans notre cas, la base de données (BDD) est le cœur de notre projet et de l'application. Son bon fonctionnement est primordial pour permettre une bonne utilisation de l'application. C'est pourquoi nous avons consacré beaucoup de temps sur sa conception et surtout sur sa réalisation.

A la base, nous avons eu accès aux tables de la BDD du site utilisé par le client (`lmb.monocyle.info`) avec lequel l'application doit communiquer au niveau de l'import et de l'export des données. Le modèle de base de données proposé a été réalisé après plusieurs itérations lors de la conception du site web lui-même, l'administrateur du site nous ayant informé d'entrée que son schéma possédait certains défauts de conception. Du fait que nous devions communiquer avec la base de données du site web nous avons choisi de conserver le schéma proposé. De ce schéma, nous avons fait une rétro conception afin d'obtenir le schéma relationnel que nous avons utilisé pour notre base de données. Le schéma est présenté dans l'annexe D.
Une fois la partie théorique réalisée, il nous fallait trouver un modèle d'implémentation, et nous n'avions que très peu de connaissances sur les schémas utilisés pour la réalisation d'une base de données. Par conséquent, nous avons cherché des exemples de réalisation et nous avons découvert le patron de conception MVC (modèle, vue et contrôleur). L'implémentation la plus connue du modèle MVC, et que nous avons utilisé, est l'utilisation du modèle DAO (Data Access Object). Celui ci est utilisé dans la plupart des applications web dans le cadre d'utilisation de bases de données. Le principe du modèle DAO est expliqué dans la partie 3.2 du cahier des charges.
Ainsi, nous avons commencé à coder la partie modèle. Le modèle permet de représenter les objets au niveau du code. Ici chaque objet est représenté par une classe en Java. Le modèle comporte ici des entités tels que les matchs, les joueurs et leurs statistiques et leurs requêtes associées. Dans cette étape de conception, nous avons dû à plusieurs reprises changer les liens entres les joueurs et leurs statistiques et même le lien entre les statistiques elles même. En effet, on parlait parfois d'héritage d'autre fois de lien a-un, cela dépendait surtout de la façon dont la base de données allait être gérée. Finalement le modèle établi consiste à créer un objet à partir d'une table de la base donnée, en insérant dans l'objet les attributs de chaque table. Ainsi un objet `Poule` possède les attributs :

- `int id`
- `int phase_poule_id`
- `int etat`
- `int points_victoire`
- `int points_nul`
- `int points_defaite`
- `String libellé`

Ensuite, il y a eu la conception du contrôleur, qui permet de communiquer entre la partie interface utilisateur (la vue) et la BDD. Suite à nos TD de XML, nous avions un exemple d'implémentation de contrôleur. L'implémentation qui nous avons pris en exemple utilisait JDBC, qui permet de communiquer avec la base de données depuis du code Java. Nous avons donc suivi l'exemple pour tester rapidement nos modèles dans le but de les corriger en cas de dysfonctionnement. Une fois nos modèles fonctionnels et nos tests sur JDBC réussis nous avons essayé d'intégrer ce qui avait été fait à Android. Malheureusement l'utilisation de JDBC n'est pas compatible avec Android, qui implémente sa propre classe pour communiquer avec la base de données. Il nous a fallu redéfinir le contrôleur plusieurs fois avant d'avoir quelque chose de fonctionnel pour Android. Cela a eu comme conséquence de nous retarder lors de la présentation du premier livrable.
L'implémentation du finale du contrôleur avec Android est la suivante, en prenant comme exemple l'utilisation sur l'objet `Equipe`.

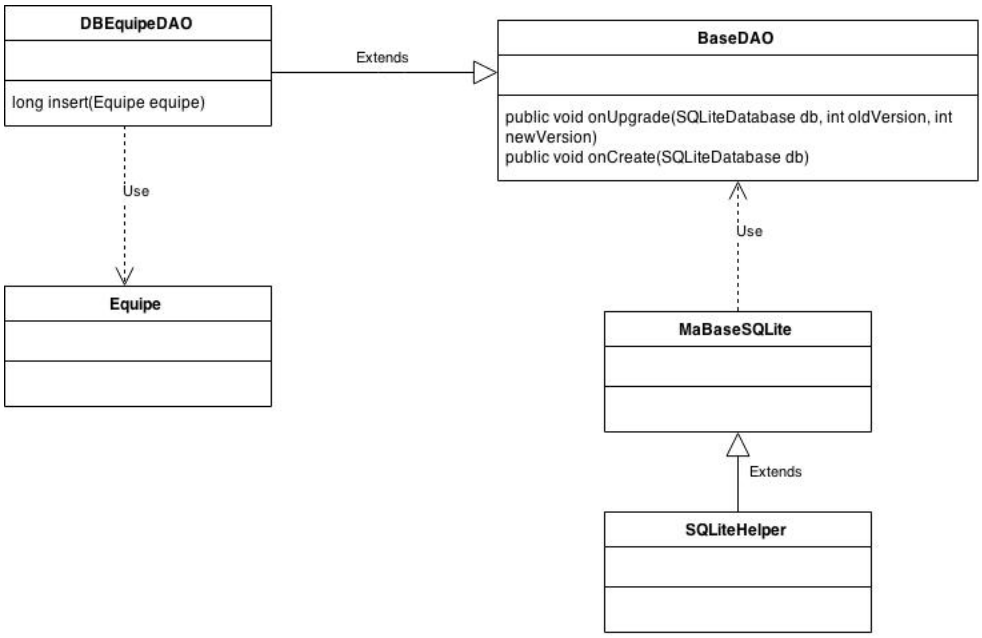


FIGURE 3.5 – Exemple de fonctionnement avec l'entité `Equipe`

Le problème posé par cette implémentation et qu'il n'est pas possible d'avoir une seule instance de contrôleur qui permet de gérer l'accès à tous les objets de la base de données. Notre solution nous contraint à instancier un contrôleur par objet présent dans le modèle. Ce qui s'avère coûteux au niveau du nombre de fichiers créés et en termes de nombres de connexions à la base de données réalisées lorsque plusieurs objets doivent être accédés dans la base. En effet la classe `BaseDAO` permet de réaliser les accès à la base de données qui est une instance de `MaBaseSQLite`. Comme chaque fichier de DAO implémente `BaseDAO` plusieurs connexions sont effectuées pour écrire dans la base de données.

Ce qui diffère de ce qu'on a pu avoir avec nos modèles de base. Cependant, Pour le traitement des statistiques nous avons de petites subtilités. Nous partons d'une classe `Stat` qui contient l'identifiant d'un joueur dans la BDD et les statistiques du joueur en question.
Une `Stat` s'obtient soit grâce à des méthodes tels que `getStatFromJoueur(Joueur j, int`

`matchId)` lorsqu'il s'agit d'extraire les données dans la BDD, soit elle est créée et modifiée lors d'une saisie pendant un match. Ainsi, lors d'une requête dans la BDD la classe `Stat` "se divise" sur des classes tels que `Shoot` ou `Fautes` à travers les classes `DBShootDAO` ou `DBfauteDAO`. La classe `Stat` est très utile pour des réalisations telles que l'import/export où toutes les données sont stockées dans un seul endroit.

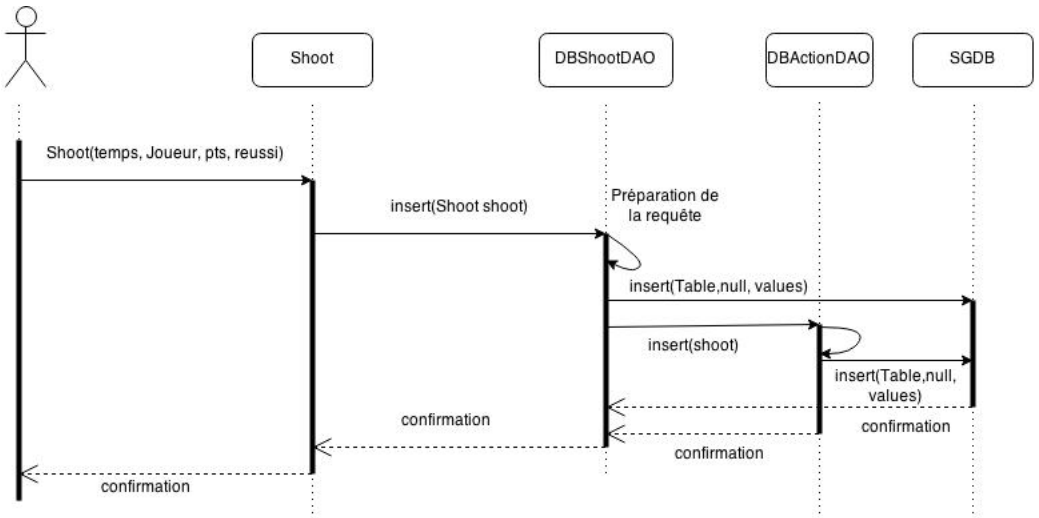


FIGURE 3.6 – Saisie d'une statistique

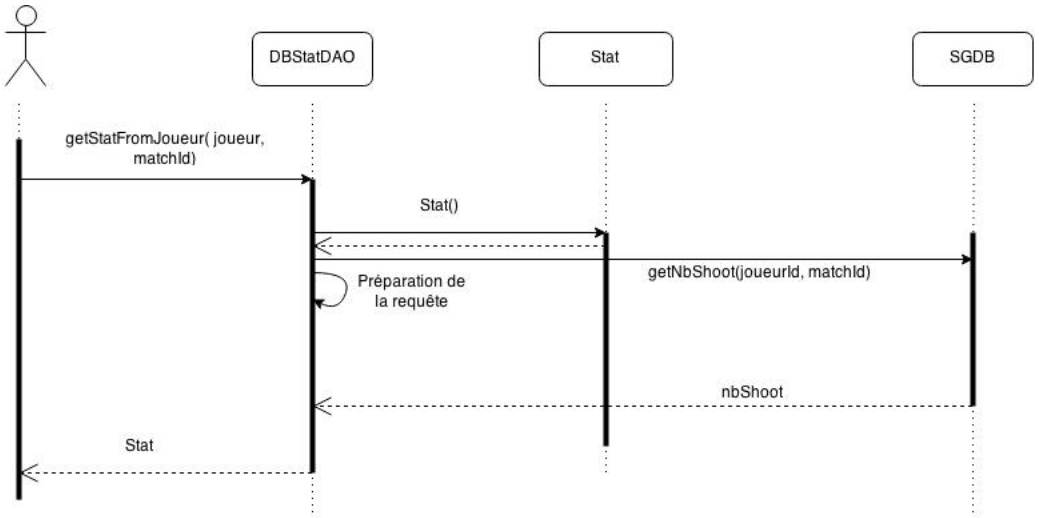


FIGURE 3.7 – visualisation d'une statistique (simplifié)

3.3 Format d'import et d'export des données

3.3.1 Entre le site et l'application

L'application doit permettre d'importer et d'exporter les données qui sont destinées à être ajoutées sur le site `http://lmb.monocycle.info/`. Nous avons dans un premier temps eu le choix de traduire les données en document XML ou JSON. Ces deux formats permettent un échange simple d'informations, compréhensible facilement par un humain de par leur structure sous forme d'arbre (avec des balises imbriquées pour le XML et des objets pour JSON). Les fichiers XML et JSON sont également simples à traduire par l'ordinateur puisqu'il existe des fonctions dans différents langages permettant de parser ces fichiers. De plus, nous avions déjà eu l'occasion d'utiliser le XML dans le cadre de travaux dirigés effectués durant le semestre 7, nous y étions donc déjà familiers.

Nous avons finalement opté pour l'utilisation de JSON. En effet, un document JSON représentant un objet, il est plus facile à interpréter avec un langage de programmation orienté objet tel que Java, à l'opposé d'un document XML dont la syntaxe est plus lourde et dont l'utilisation des méthodes de DOM (Document Object Model) peuvent être plus lentes. Ce qui a son importance notamment avec une application Android où l'optimisation des ressources est une problématique plus importante que pour un ordinateur.

Nous avons ensuite pris contact avec l'administrateur du site pour savoir ce qu'il était intéressant de pouvoir importer et exporter et aussi pour poser des questions concernant sa base de données de manière à synchroniser les données entre les deux bases. En effet, si notre modèle relationnel est inspiré du sien, il comporte tout de même quelques différences, notamment dans le choix de certaines constantes. Par exemple, la table **Match** comporte un champ résultat : nous avons choisi d'y insérer l'identifiant de l'équipe du vainqueur et 0 en cas de match nul tandis que l'administrateur utilise 1 pour une victoire de la première équipe, 2 pour la deuxième et 3 pour un match nul.

Concernant l'import des données, nous avons choisi de pouvoir importer des équipes ainsi que des joueurs. Ceci s'effectue au tout début d'un tournoi pour pouvoir récupérer tous les différents acteurs du tournoi, avec tout de même la possibilité de créer des équipes et des joueurs dans l'application et donc de les ajouter au tournoi.
Une des difficultés rencontrées provient du fait que les identifiants des différentes tables de la base de donnée interne à l'application ne sont pas les mêmes que ceux du serveur, ce qui pose également un problème pour les clés étrangères et donc pour lier correctement les différentes tables. Une solution à laquelle nous avons pensé et que l'on a choisi a été de rajouter un champ **id_serveur** dans les tables de notre base «concernées» par l'import/export, c'est à dire les tables **Equipe** et **Joueur**. Lorsqu'une équipe ou un joueur est créé directement dans l'application, ce champ est égal à -1. De cette manière, la distinction est claire et les informations du serveur ne seront pas perdues lors de l'export des données. Pour le problème des clés étrangères, le lien entre les tables est mis en évidence dans l'arborescence du fichier JSON : un fichier d'import dont un exemple du format à respecter est donné dans le manuel d'utilisation est constitué d'un tableau d'équipes où chaque élément du tableau est un objet représentant une équipe qui contient lui-même un tableau de joueurs. De cette manière, le lien entre les entités est clair et ne nécessite pas de connaître les clés étrangères.

L'export comporte les données liées à un match qui a été joué et dont on veut récupérer les résultats pour les mettre ensuite sur le site. Il s'agit d'afficher les informations de la table **Match** ainsi que les différentes actions avec les temps de jeu, et les numéros des joueurs pour le match

donné. Là encore se pose le problème des clés étrangères liés au temps de jeu, à la table **Action** et au type réel de l'action. De la même manière que pour l'import des données, on met en évidence les liens avec l'arborescence du fichier : chaque objet équipe (équipe A et équipe B) contient d'une part un tableau d'actions où chaque élément est un objet qui contient les trois objets types d'action, action et temps de jeu, et d'autres part un tableau de formations qui contient les informations de la table **formationJoueur**, en d'autre termes l'identifiant du joueur sur le serveur et le numéro qu'il a porté durant la rencontre.

3.3.2 La feuille de match

En plus d'une synchronisation des BDD, il fut intéressant pour le client de pouvoir produire des feuilles de match au format PDF. Encore une fois ici nous avons eu à apprendre sur le terrain la méthode pour répondre à ce besoin sachant que ce fut la dernière fonctionnalité implémentée. Par conséquent, elle a été réalisée dans les derniers jours précédant le rendu du projet.
Pour ce travail, nous avions le choix d'utiliser le paquetage iText qui créer un .pdf formaté par l'intermédiaire d'un code Java (cf exemple de feuille de match en annexe). Malgré l'existence d'un paquetage Android qui répond à ce besoin, il nous a semblé plus rapide d'utiliser iText et d'inclure les .jar qui étaient associées dans l'application à cause du manque de temps. Ce qui a pour conséquence d'alourdir un peu plus le poids de l'application.

Néanmoins, le pdf peut être encore amélioré. En effet, la mise en page est minimale et il serait intéressant, si une amélioration à lieu de rajouter des informations tel que le détails des périodes de jeu, et les temps de jeu (si implémentation il y a).

3.4 Conception de l'interface

La conception de l'interface a constitué une phase clé du projet car l'intégralité du fonctionnement côté client se résume à l'utilisation de l'interface. Cette phase de conception a donc été importante afin d'obtenir une application simple à utiliser pour le client, respectant des critères d'ergonomie et étant en accord avec les fonctionnalités exigées par le cahier des charges. Nous allons exposer dans cette partie les réflexions qui ont eu cours lors de la conception.

3.4.1 Méthode de fonctionnement

Pour développer l'application nous avons suivi une méthode de développement par cycle itératif. Nous avions rendez-vous avec le client environ une fois par mois. Cela nous permettait à chaque fois de proposer une nouvelle version de l'application. Chaque nouvelle version devait prendre en compte les remarques du client sur la version précédente afin d'effectuer les modifications nécessaires. De ce fait de nouvelles fonctionnalités et de nouveaux besoins sont logiquement venues s'additionner à ce qui avait été prédéfinies dans le cahier des charges. Pour concevoir chacun des morceaux de l'interface, le mode de fonctionnement que nous avons suivi peut être décrit de la façon suivante :

Etape 1 : Définition des besoins
Etape 2 : Proposition d'une interface au client à l'aide d'un dessin papier accompagné d'explications sur les fonctionnalités qui seront implémentées
Etape 3 : Echange avec le client
Etape 4 : Réalisation de l'interface
Etape 5 : Retour du client sur ce qui a été fait

Etape 6 : Apport de correctif

3.4.2 Interface de match

L'interface de match peut être considérée comme le noyau de l'application. En effet comme nous l'avons vu plus haut c'est elle qui fait défaut aux applications de saisie de statistiques existantes, notamment du fait que la saisie de statistiques est trop lente. Pour la conception de cette interface nous avons dès le début du projet été en relation avec le client afin de réfléchir à une interface qui réponds aux besoins exprimés. C'est la partie du projet qui a demandé le plus grand nombre d'itérations pour sa conception. Les besoins exprimés pour l'interface sont définis dans le cahier des charges dans la partie **Saisie des données**. La première ébauche de cette interface (figure 3.6) est celle présentée dans le cahier des charges.

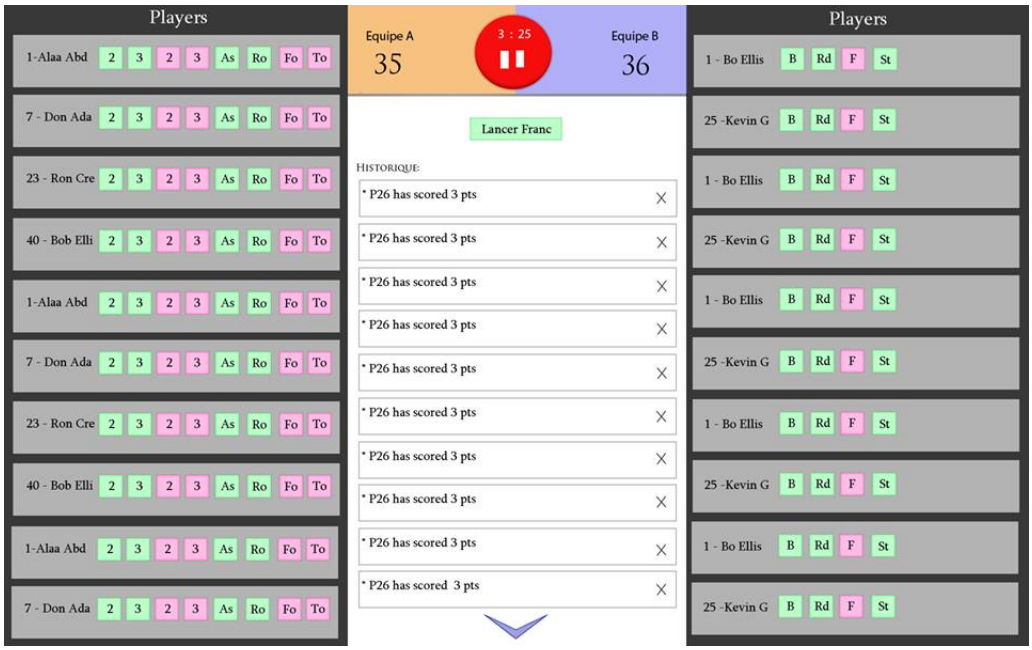


FIGURE 3.8 – Première proposition d'interface

Sur cette première version plusieurs choses n'allaient pas comme :

- La taille des boutons qui était trop petite
- L'historique qui prenait trop de place
- La non présence du JOUEUR ÉQUIPE
- Pas de bouton SWITCH pour changer la possession
- L'équipe en possession du ballon n'est pas indiqué
- Le fait qu'il n'y ai que deux couleurs de boutons est dérangeant pour l'utilisateur
- Un seul type de faute est indiqué
- Un seul bouton en accès deux clics
- Pas de distinction faite selon la couleur de l'équipe
- Impossible de prendre en compte des numéros de maillot étant des chaînes de caractères

De plus du fait des difficultés de programmation que nous avons pu rencontrer sur Android, la disposition du chronomètre, des scores des deux équipes a été amené à évoluer. De même le

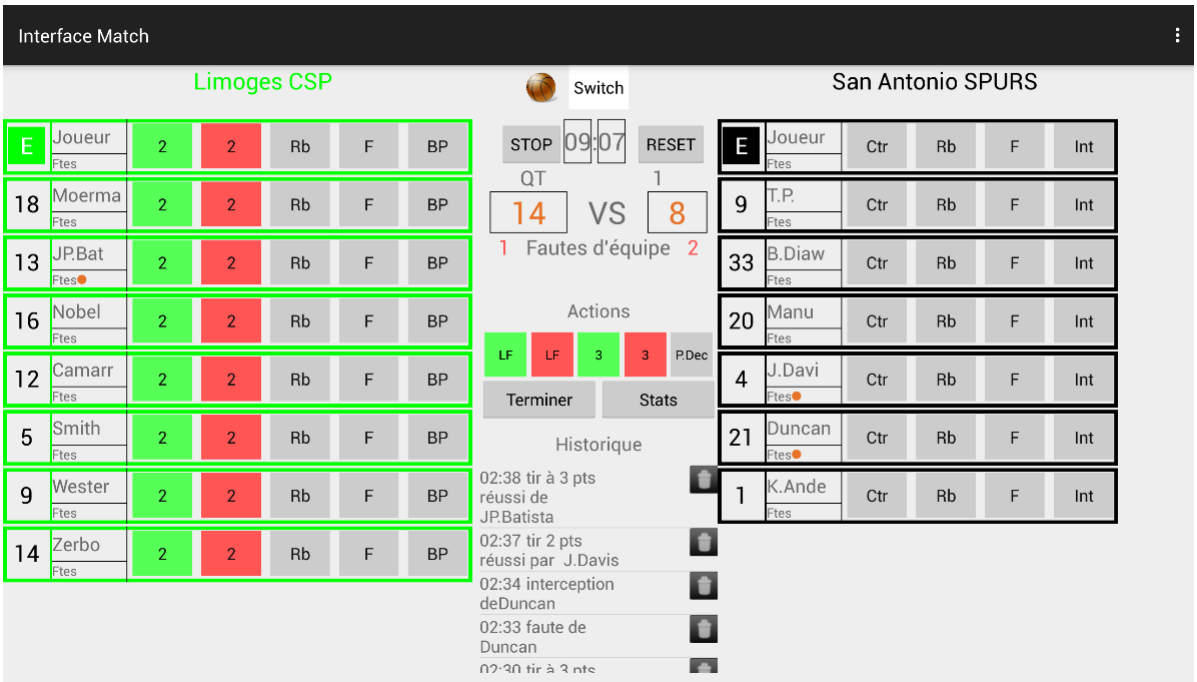


FIGURE 3.9 – Deuxième proposition d'interface

fait d'avoir des boutons plus gros pour la saisie nous a contraint à avoir un nombre de 5 boutons maximums accessibles en accès rapide. Ainsi après quelques réflexions nous avons proposé sur l'interface, qui après plusieurs correctifs minimes, est devenu l'interface finale obtenue :

3.4.3 Lancement d'un match

L'interface permettant de démarrer un match n'avait de fonctionnalités précisément défini dans le cahier des charges. Certaines étaient évidentes et d'autres sont venues s'additionner suite aux échanges que nous avons eu avec le client. Globalement nous n'avons eu à ajouter que peu de fonctionnalités par rapport à la proposition qui avait été faite au client qui est en annexe la suivante :

Choix des équipes

L'écran de choix des équipes est celui situé en haut à gauche. Il permet de répondre aux besoins suivants.

— Sélectionner deux équipes différentes dans la base de données
— Ajouter de nouvelles équipes
— Choisir une couleur de maillot pour chaque équipe
— Définir la durée du match (nombre de périodes et durée de chaque période)

Ajout/Suppression de joueurs

Une fois le choix des équipes effectué, nous permettons à l'utilisateur de choisir les joueurs qu'il souhaite ou non ajouter à son équipe. Les besoins définis sont les suivants :

— Possibilités d'ajouter un nouveau joueur à une des deux équipes
— L'ajout d'un joueur peut être effectuer à partir d'un joueur existant de la base de données (demande du client)

— Supprimer un joueur de l'équipe

Sélection des joueurs

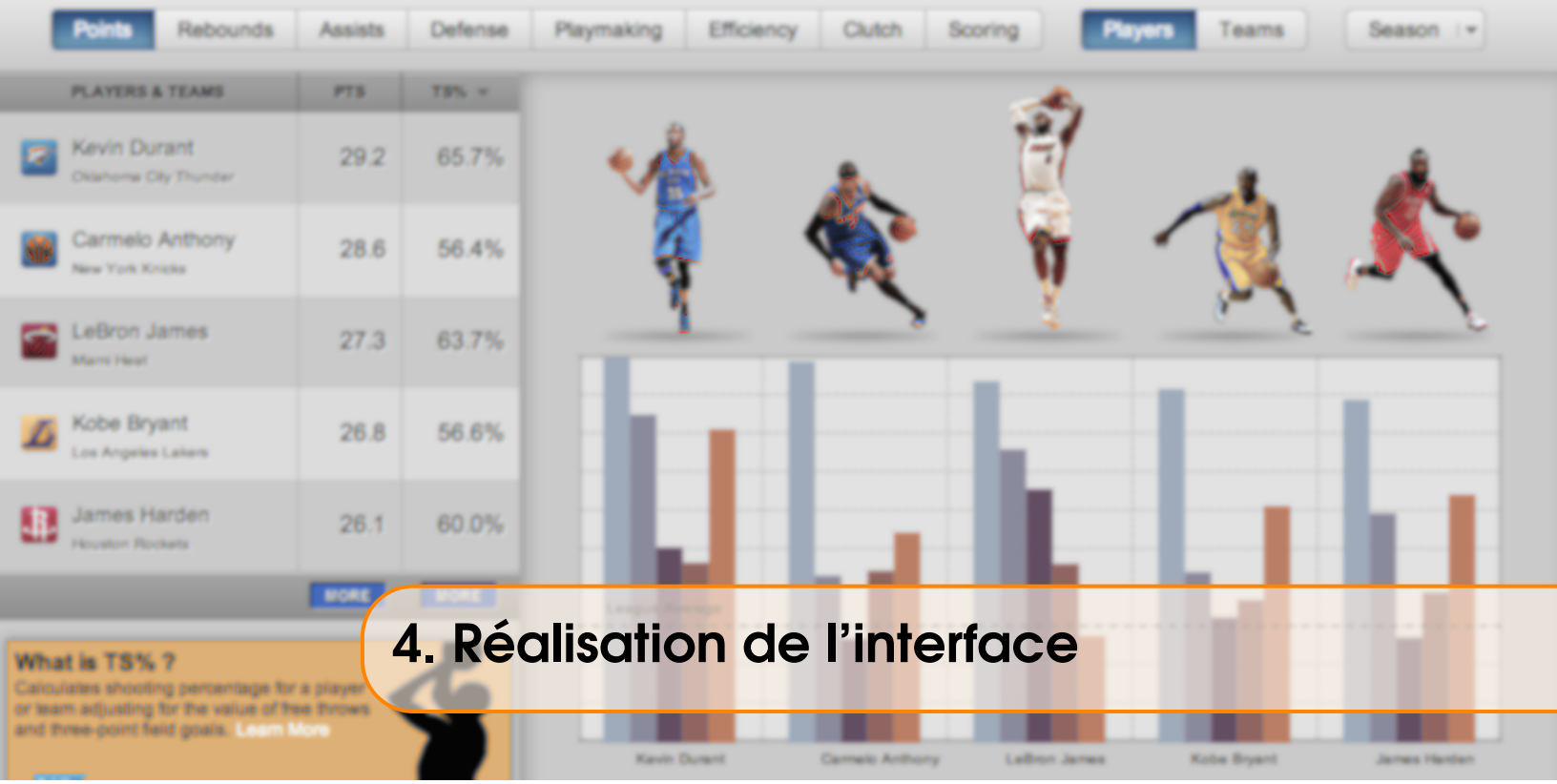
Une fois l'appartenance des joueurs à une équipe bien défini, il ne reste plus qu'à les sélectionner pour le match à venir et à définir le numéros de maillot de chacun des joueurs. La sélection du joueur se fait par un clic sur le joueur et la définition du numéro de maillot d'un joueur se fait par un clic sur le numéro qui déclenche l'apparition d'une boîte de dialogue. La boîte de dialogue permettait seulement de saisir des nombres entiers au départ, mais le client nous a demandé de pouvoir sélectionner plutôt des chaînes de caractères. Il a également demandé à garder en mémoire la dernière sélection, option à laquelle nous n'avions pas pensé. Les fonctionnalités finales sont donc les suivantes :

— Proposer par défaut la dernière sélection utilisée pour l'équipe
— Sélectionner ou déselectionner les joueurs de l'équipe
— Définir un numéro de maillot pour chaque joueur

3.4.4 Interface de navigation d'un tournoi

La conception de l'interface de navigation d'un tournoi a été complexe du fait que nous avions un modèle d'interface pour ordinateur (`lmb.monocyle.info`) proposant de nombreuses fonctionnalités, mais que nous devions nous adapter au format tablette qui propose un écran plus réduit. La difficulté est de synthétiser les fonctionnalités disponibles de manière à ce que l'utilisation de l'interface par l'utilisateur reste instinctive et ergonomique. Pour cette partie du projet nous avons fait donc confiance à notre créativité, tout en essayant de mettre en place ce que nous avions appris en cours d'interface homme/machine. Les premiers dessins de l'interface de tournoi sont disponibles en annexe C.

Ces dessins étaient des ébauches et au fur et à mesure de l'implémentation et des contraintes que nous avons pu rencontrer sur Android, l'interface a été amené à évoluer. C'est notamment le cas de la conception pour les phases tableaux, nous voulions au départ afficher un arbre de tournoi comme fréquemment rencontré mais cela n'a pas été possible et nous avons du trouver un autre moyen. D'autres parties s'adaptaient plutôt bien à une réalisation d'Android notamment le menu latéral, qui est fréquemment utilisé pour des application Android et l'interface de navigation pour une phase de poule, où l'utilisation de fragments est instinctive pour réaliser ce qui est proposé. On met en annexe les dessins, on parle des idées qu'on a eu.



4. Réalisation de l'interface

Dans cette partie, nous allons vous présenter la façon dont nous avons construit l'application en utilisant le modèle de base de données présenté dans la partie 3.2. L'objectif ici n'est pas de présenter le code que nous avons écrit mais la réflexion que nous avons eu pour agencer les différents morceaux de l'application et les faire interagir. Cette partie sera également l'occasion de comparer les résultats obtenus pour les écrans par rapport à ce qui a été défini dans la partie conception.

En résumé quelqu'un qui connaît un peu Android et qui veut refaire la même application doit comprendre la manière dont nous avons procéder pour générer notre code.

4.1 Lancement d'un match

Pour permettre à l'utilisateur de démarrer un nouveau match, trois écran se succèdent pour enregistrer les informations nécessaires. Chaque écran étant géré par une activité, nous empilons les trois activités plus l'activité de l'interface match. Lorsque le match est terminé, l'activité d'interface match envoie un signal qui indique aux trois activités de se terminer. Cela permet en particulier d'éviter que l'utilisateur ait à appuyer sur **retour** lui-même pour supprimer les activités.

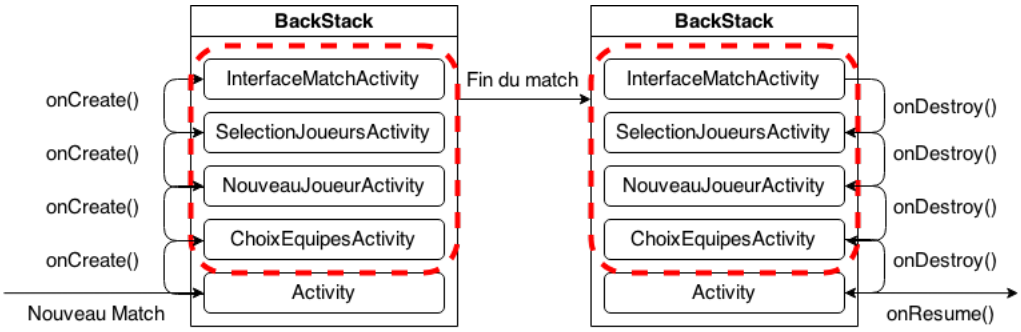


FIGURE 4.1 – Évolution de la pile des activités pendant le lancement d'un match

4.1.1 Sélection des équipes

L'écran de sélection des équipes est présenté dans la partie 3.1 de l'annexe B.

4.1.2 Possibilités d'ajout de nouveaux joueurs

Afin de proposer les fonctionnalités définies dans la partie 3.4.3, nous avons du définir à partir de quand un joueur est considéré comme appartenant à une équipe ou non. Nous avions pensé tout d'abord à chercher l'intégralité des joueurs ayant jouer au moins un match pour cette équipe. Mais nous avions peur que cela fasse trop de joueurs à proposer au bout d'une certaine durée d'utilisation. De plus cela n'aurait pas permis de supprimer les joueurs au niveau de l'affichage et de la base de données simultanément. Nous avons donc choisi de créer une formation par défaut dans la table **Formation** qui est propre à chaque équipe. Nous avons ajouter l'attribut `TYPE_FORMATION` à la table **Formation** pour repérer la formation par défaut. La formation par défaut est initialisée lors de la création d'une nouvelle équipe. Obtenir la liste des joueurs de l'équipe revient ainsi à chercher les joueurs appartenant à la formation par défaut dans la base de données. L'ajout ou la suppression de joueur à la formation par défaut se fait par l'intermédiaire de l'écran de sélection des joueur (partie 3.2 de l'annexe B). Avec cette réalisation l'utilisateur défini lui-même les joueurs qui appartiennent à l'équipe.

4.1.3 Sélection des joueurs pour un match

L'écran de sélection des joueurs est présenté dans la partie 3.2 de l'annexe B. L'activité de sélection des joueurs doit proposer par défaut la dernière sélection enregistrée dans la base de données. Pour cela nous avons défini un nouveau type de formation avec le type `LAST` pour l'attribut `TYPE_FORMATION`. Ensuite mis en place un processus particulier à chaque lancement d'un nouveau match :

1. Création d'une nouvelle formation dite classique dans la table **Formation**.
2. Récupération de la dernière formation utilisée. On utilise l'attribut `TYPE_FORMATION` qui vaut `LAST`, et on supprime la formation de la base de données ainsi que toutes les tables `FormationJoueur` qui lui sont liées.
3. On inserts une nouvelle formation de type `LAST` et on lui associe tous les joueurs sélectionnés pour le match.
4. On effectue la même opération d'association avec la formation classique précédemment insérée.

Ce processus maintient à chaque fois une unique formation de type `LAST` en supprimant et créant une nouvelle sélection à chaque match. La formation classique permet quand à elle de référencer et garder en mémoire la formation utilisée pour le match, un match ayant dans ses attributs un `FORMATION_ID`.

L'écran présente également un compteur qui impose à l'utilisateur de sélectionner au moins 5 joueurs dans l'équipe. Cependant il nous a explicitement été demandé de pouvoir lancer un match avec moins de 5 joueurs dans l'équipe (voire 0). Dans le cas où 0 joueur sont sélectionnés, seul le joueur équipe apparaîtra sur l'interface match. Une boîte de dialogue s'affiche lorsque moins de 5 joueurs sont sélectionnés mais celle-ci ne bloque par le lancement du match.

4.2 Navigation dans un tournoi

4.2.1 Accueil Tournoi

L'écran d'accueil est une activité basique qui contient 3 boutons. dont les fonctionnement sont les suivants :

— `Nouveau Tournoi:` Lance l'activité qui gère la création d'un nouveau tournoi
— `Reprendre Tournoi:` Permet de sélectionner un tournoi parmi la liste des tournois de la base de données
— `Supprimer Tournoi:` Permet de sélectionner un tournoi et de le supprimer de la base de données.

Pour obtenir la liste des tournois de la base de données une requête `getAll()` est appelée sur la table **Tournoi**. L'activité implémente également la méthode `onResume()` qui permet de mettre à jour la liste des tournois proposés à l'utilisateur dans le cas où l'on revient à cet écran après la création d'un tournoi.

4.2.2 Nouveau Tournoi

L'activité de création d'un nouveau tournoi demande juste à l'utilisateur de rentrer un **nom** et un **lieu** pour le tournoi. Une fois que l'utilisateur clique sur valider, un nouvel objet **Tournoi** est créé puis inséré dans la base de données. On récupère alors l'identifiant du tournoi dans la base pour le passer en paramètre de la prochaine activité, qui va permettre de sélectionner les équipes pour le tournoi.

4.2.3 Sélection des équipes pour le tournoi

Cet écran est visible dans la partie 5.1 de l'annexe B.

Pour cette activité une requête `getAll()` est appelée sur la table **Equipe** afin de récupérer la liste des équipes disponibles dans la base de données. Néanmoins l'utilisateur doit pouvoir ajouter de nouvelles équipes. Pour cela deux choix sont possibles : soit ajouter une nouvelle équipe, soit en importer une nouvelle à partir d'un fichier `.json`. L'appel à l'une des fonctions démarre une nouvelle activité qui, une fois son exécution terminée, renvoie un résultat défini de la façon suivante :

— `resultat == RESULT_OK` si la création de l'équipe s'est correctement déroulée.
— `resultat == RESULT_CANCELED` sinon.

Dans le cas où l'activité reçoit le résultat `RESULT_OK`, on effectue une mise à jour de la liste des équipes disponibles dans la base de données. Nous allons maintenant présenter plus en détails les deux activités d'ajout d'une nouvelle équipe.

Ajout d'une équipe

Cette fonction est présentée dans la partie 5.1 de l'annexe B

Import d'une équipe

La fonction d'import d'équipe est appelée en interne dans l'activité une fois le fichier `.json` sélectionné.
Une fois que la sélection des équipes pour le tournoi est faite, on récupère l'identifiant de chaque équipe afin de la lier au tournoi en faisant une insertion dans la table **TournoiEquipe**. On démarre ensuite l'activité de navigation du tournoi en prenant soin de supprimer l'activité précédente de la pile des activités.

4.2.4 Menu de navigation

Vue d'ensemble

La vue d'ensemble permet de voir les informations importantes du tournoi.

Classement

Cet activité propose à l'utilisateur d'enregistrer un nouveau classement en sélectionnant la position de l'équipe dans le classement du tournoi, ainsi que le nombre de points attribués au classement. Les équipes proposées dans la sélection sont toutes les équipes participants au tournoi. Celles-ci sont récupérées via la requête `getTeamsList(int tournoiId)` appelée sur la table **EquipeTournoi**. Une fois le classement enregistré le nombre de points peut être modifié. Néanmoins une modification de toute la ligne n'est pas possible, la meilleure méthode de modification proposée est une suppression du classement et la création d'un nouveau.

4.2.5 Création d'une phase

Il est possible à partir du menu de tournoi de créer une nouvelle phase : phase de poule ou phase tableau. Les deux cas sont similaires et la procédure suivie est la même :

1. Création de la phase avec insertion d'un nouvel objet phase dans la base de données
2. Choix du type de phase
3. Renseignements sur les options de la phase
4. Sélection des équipes participants à la phase

La difficulté de cet enchaînement est de gérer correctement le cas où l'utilisateur souhaiterai revenir en arrière et modifier les informations enregistrées.

4.2.6 Navigation poule

Le résultat de l'implémentation est présenté dans la partie 5.4 de l'annexe B.

L'écran de navigation pour une phase de poule a été plus complexe à implémenter. Nous avons dû utiliser les fragments pour pouvoir gérer plusieurs poule avec une seule activité. Nous avons schématisé ci-dessous le mode de fonctionnement.

Comme expliqué dans la partie 3.1.3 une activité peut gérer plusieurs fragments. Dans notre cas pour une phase de poule donnée, l'activité peut récupérer les poules de la phase depuis la base de données et les enregistrer dans la liste `pouleList`. À partir de cette liste, l'on cherche à créer pour chaque poule une vue **classement** et une vue **calendrier**. On créer associe donc à chaque poule un couple de fragments `ClassementPouleFragment` et `CalendrierPouleFragment`. Au lancement de l'activitée (méthode `onCreate()`), on instancie autant de couple de fragments qu'il y a de poules. Une fois tous les couples de fragments instanciés, l'activité choisie quel couple elle souhaite affiché à l'écran via le `FragmentManager`. Afin de communiquer avec les fragments et de pouvoir changer dynamiquement de vue, l'activité doit implémenter des méthodes de callback. Ici **NavigationPouleActivity** implémente les deux méthodes de de callback `onNextPoule()` et `onPreviousPoule()` du fragment de classement. Ces méthodes sont déclenchées lorsque l'utilisateur appuie sur une flèche pour accéder soit à la poule suivante ou à la poule précédente. Chacun des fragments possède aussi des méthodes `update()` publiques qui permettent à l'activité de mettre à jour les informations contenus dans le fragment, et par conséquent les informations affichées. Les méthodes `update()` sont appelées lorsqu'un match lancé depuis le calendrier est terminé.

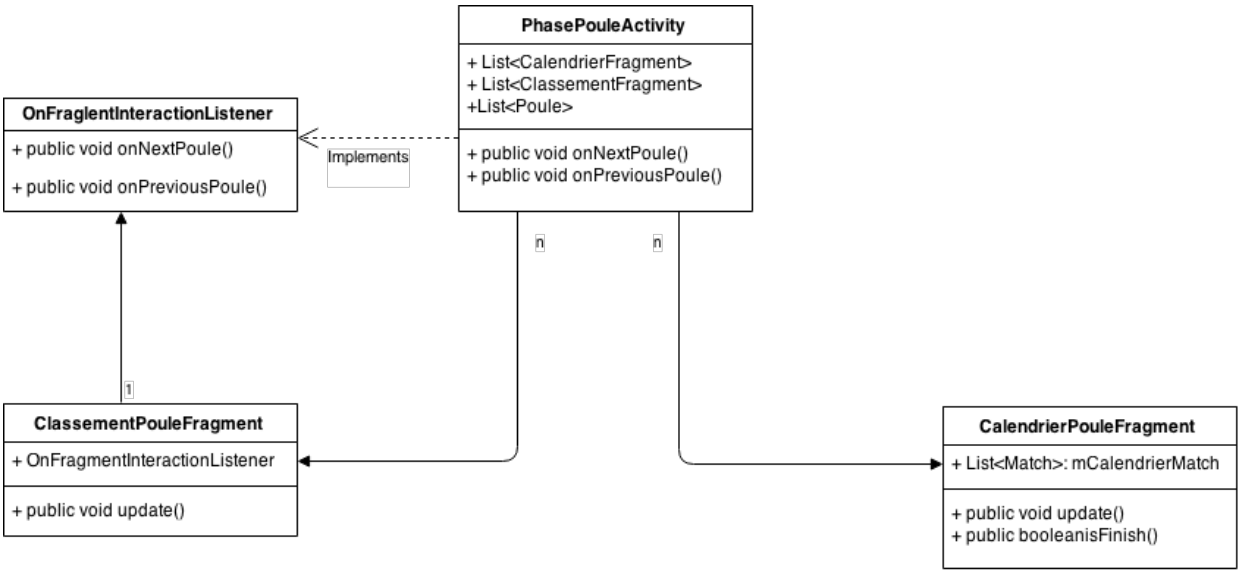


FIGURE 4.2 – Diagramme de classe pour l'activité **NavigationPouleActivity**.

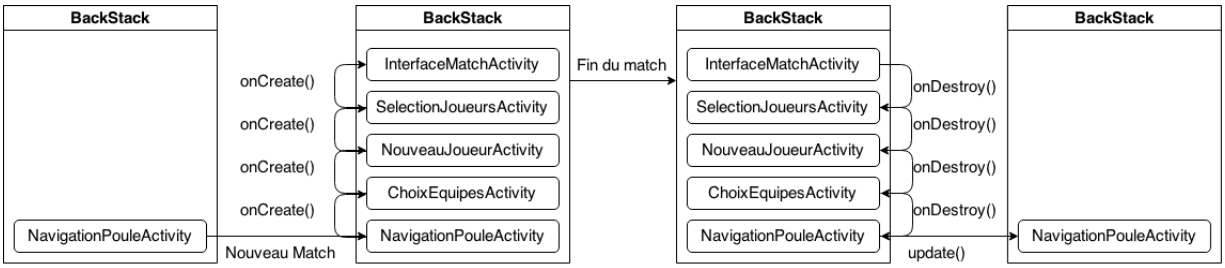


FIGURE 4.3 – Pile des activités lors du lancement d'un nouveau match.

4.2.7 Navigation tableau

Le principe de fonctionnement de l'activité gérant une phase tableau est similaire à celui utilisé pour une phase de poule. Contrairement à la phase de poule la phase tableau n'utilise qu'un type de fragment : le fragment calendrier. Le fragment de calendrier est un copier coller de celui utilisé pour les phases de poule, avec un ajout de plusieurs fonctionnalités indispensables :

— Définir les deux équipes à confronter pour un match
— Gérer dynamiquement la proposition des équipes en fonction des victoires et défaites lors du tour de final précédente. Par exemple on veut que seules les équipes ayant gagné leur quart de finale puissent jouer les demi finales.

4.3 Interface de match

L'interface match a déjà été présentée dans la partie 3.4.2. L'interface de match est décomposée en 4 parties principales : la barre supérieure située en haut de l'écran, les 2 modules latéraux situés de part et d'autre de l'écran séparés par le module central.

4.3.1 Module 1 : Barre supérieure

Le premier module de l'interface de match est la barre supérieure. Celle ci contient les noms de 2 équipes participant au match repartis de chaque coté de l'écran. Au milieu de cette barre se trouve un bouton SWITCH permettant de modifier la possession de la balle ainsi qu'un indicateur permettant de voir quelle équipe a la possession.

4.3.2 Modules latéraux : Joueurs et saisie des statistiques

Les modules latéraux permettent la saisie des statistiques lors du match. Ce module consiste en une liste de joueurs. A chaque ligne de cette liste est associée une série de boutons correspondant à différentes actions : 2 points réussi, 2 points raté, Rebond, Interception etc. Les actions présentes sur ces boutons sont des actions à accès instantané, c'est à dire qu'elles doivent pouvoir être comptabilisées en un seul clic. Ce module est présent à l'identique de chaque coté de l'écran (un pour chaque équipe). Le contenu de ce module dépend donc de la possession de la balle. Si le module est située du coté de l'équipe offensive, les boutons présents seront donc les boutons liés aux actions d'attaque comme par exemple les tirs et les rebonds offensifs. Au contraire, si le module est situé du coté de l'équipe défensive, les boutons présents seront ceux liés aux actions défensives comme par exemple les blocs, rebonds défensifs ou bien les fautes défensives.

Chaque module latéral est modifié en fonction de la couleur de l'équipe. Cela permet d'apporter une aide visuelle pour le statisticien qui lui permettra d'identifier plus rapidement l'équipe qui effectue les actions.

4.3.3 Module central

Chronomètre

Le module central est séparé en plusieurs parties. la première partie située dans la partie supérieure est occupée par le chronomètre. Celui ci mime un afficheur tels que ceux présents dans les gymnases. On y trouve donc le temps, le numéro de période, le score de chaque équipe ainsi que le nombre de fautes de chaque équipe. Son fonctionnement est détaillé dans la partie 4.5 du manuel d'utilisation.

Action

En dessous du chronomètre se situe dans une rangée de boutons permettant de comptabiliser des actions en 2 clics. Ces actions sont accessibles plus lentement car elles ont une fréquence d'apparition plus réduite que d'autres actions. Intégrer ces actions dans les modules latéraux aurait rendu l'interface surchargée et le manque de place empêche également de placer toutes les actions dans les modules latéraux. La prise en compte d'un de ces actions s'effectue en deux temps. D'abord, on sélectionne l'action voulue. Par la suite, selon le type de l'action (offensive ou défensive), l'équipe pour laquelle peut être prise en compte l'action reste affichée dans le module latéral tandis que l'autre est masquée jusqu'à ce qu'un joueur soit sélectionné.

Historique

La dernière partie du module central est l'historique. A chaque fois qu'une action est ajoutée, une ligne est ajoutée dans la liste de l'historique avec l'heure à laquelle l'événement s'est déroulé, le type de l'action et son acteur. Un bouton supprimer est également présent à l'extrémité droite de la ligne afin d'annuler la saisie de l'action.

4.3.4 Améliorations de l'interface

L'interface de match peut bien évidemment être améliorée. Premièrement, au niveau visuel on pourrait imaginer une meilleure intégration de la couleur des équipes et des numéros de maillot des joueurs dans l'interface. De même, on peut imaginer pouvoir ajouter manuellement dans l'historique des actions à un temps spécifié dans le match dans le cas où un événement aurait été oublié.
D'ailleurs, l'interface décrite ci-dessus est le résultat de nombreuses corrections et modifications. En effet, au début de la conception de l'interface de match, le chronomètre était minimaliste. Aussi, l'affichage de la couleur par défaut de l'équipe n'était pas présent ce qui rendait l'identification des équipes plus difficiles pour le statisticien. Ces quelques exemples illustrent le procédé d'améliorations successives que nous avons suivi pour construire cette interface.

4.4 Visualisation des statistiques

À la fin d'un match, l'écran de saisie transmet l'identifiant de celui-ci à la demande de visualisation des statistiques. Cet identifiant permet d'accéder dans la base de données à l'ensemble des joueurs et des actions de ce match.
L'écran se divise en deux fragments, regroupant chacun les données d'une des deux équipes.
Une nouvelle classe par joueur et par match, appelée `Stat`, a été crée. Elle a pour attribut chacune des statistiques qui peuvent être saisies sur l'interface match. La méthode `getStatFromJoueur(int joueurId, int matchId)` appelle les méthodes calculant le nombre de fautes `getNbFautes(int joueurId, int matchId)`, le nombre de tirs `getNbShoot(int joueurId, int matchId)`, etc...
Ces méthodes effectuent une requête `SELECT COUNT(*)` par l'intermédiaire de `rawQuery()` de la classe `SQLiteDatabase`.
Les tirs étant différenciés entre réussis et manqués, un rapport nombre de réussis sur la somme des tirs tentés est affiché. Une statistique appelée Évaluation, communément répandue, a été intégré.
L'écran de statistiques est présenté en détail dans la partie 4.7 de l'annexe B.

4.5 Historique Final

Afin de rendre disponible l'ensemble des saisies du match, un écran de l'historique final a été ajouté à la suite de celui des statistiques. Chacune des actions apparaît sous la forme suivante :

"TEMPS : MM :SS JOUEUR : NOM PRENOM TYPE_ACTION"

5. Évaluation du produit final

Malgré nos efforts afin de rester au plus fidèles au cahier des charges, l'application finale n'est pas exactement identique point par point à ce qui est présent dans le cahier des charges.

5.1 Comparaison avec le cahier des charges

5.1.1 Interface de match

Les différences les plus notables par rapport au cahier des charges se situent au niveau de l'interface de match. Tout d'abord, les boutons présents dans les modules latéraux ont étés modifiés par rapport à ce qui était prévu dans le cahier des charges. De même, d'après le cahier des charges, il doit être possible de configurer l'organisation des boutons dans l'interface de match. L'idée était d'avoir plusieurs profils différents afin d'avoir plusieurs actions différentes disponibles en accès rapide. Un profil Amateur permettrait d'avoir des actions "basiques" en accès 2 clics comme par exemple les 2 Points et les Fautes. Un autre profil Professionnel pourrait avoir les actions à 3 points par exemple. En effet, celles ci ne sont pas souvent effectuées et mettre les tirs à 3 points en accès rapide prendrait de la place inutilement dans la barre des actions rapides. Enfin, une 3e option pourrait permettre de définir un layout personnalisé afin de choisir les actions disponibles en accès rapide. Malheureusement seul le profil qualifié d'Amateur a été implémenté.
Un autre déviation par rapport au cahier des charges est la possibilité d'ajouter des actions manuellement dans l'historique. Il n'est dans notre produit final possible d'ajouter dans l'historique des actions uniquement en cliquant sur les boutons de l'interface.

5.1.2 Statistiques

La visualisation des statistiques pour un match a été réalisé conformément au cahier des charges.

5.1.3 Import/Export

Par rapport à ce qui est présenté dans le cahier des charges, l'idée principale résidant derrière l'import et l'export de données demeure la même. Cependant, le format des fichiers d'import et

d'export a été revu. Initialement, nous pensions utiliser un format XML afin d'échanger les données. Par la suite, nous avons changé ce choix pour se porter vers un format JSON, plus pratique à analyser via des bibliothèques Android. Nous pensions également pouvoir exporter l'intégralité d'un tournoi dans le cahier des charges mais nous nous sommes limités à au simple export du résultat d'un match. Concernant l'import, il est possible d'importer des équipes et des joueurs, mais aussi de déterminer le résultat d'un match sans importer toutes les statistiques relatives à ce match.

5.1.4 Gestion de tournoi

Aussi, il était prévu lors de la gestion du tournoi de pouvoir visualiser un classement des meilleurs joueurs sur le tournoi. Ce classement n'a pas été implémenté car il n'a pas été jugé comme une priorité vis à vis d'autres fonctionnalités.

BASTATS

6. Conclusion

Ce projet a été l'occasion pour chacun de nous de mener à bien un projet concret depuis la définition du cahier des charges en accord avec le client jusqu'à la livraison du produit final. C'était un première expérience sur le développement d'un projet mené dans les mêmes conditions qu'en entreprise. Et pour chacun de nous cette expérience fut positive.

Ce PFA nous a donc permis de gagner de l'expérience dans le développement de projet, et de voir à quel point il peut être difficile de s'organiser correctement au sein d'une équipe de 7 personnes. En effet il n'est pas toujours facile de bien se répartir les tâches, ou encore de communiquer correctement entre nous sur ce qui doit être réalisé. Mais la partie la plus difficile a été de concevoir l'application à partir d'un simple cahier des charges. Il a été difficile pour nous d'évaluer correctement la charge de travail nécessaire pour mener à bien l'intégralité du projet. De plus nous avons dû faire face à la découverte d'un nouvel environnement de travail qu'est Android. Cela a compliqué un peu plus la conception de l'application et l'organisation du code.

Malgré les difficultés rencontrées nous sommes satisfait du travail réalisé et du produit final proposé. En effet les fonctionnalités les plus importantes définies dans le cahier des charges ont été réalisé et les fonctionnalités manquantes n'empêche pas le bon fonctionnement de l'application.

BASTATS

A. Document des spécifications

Document des spécifications

Application de saisie de statistiques pour les matchs de basket

Lionel Adotevi Jean Barré Reda Boudjeltia Mathieu Carrié Romain Chabot
Reda Sabir Lotfi Zouad

19 décembre 2014

Table des matières

1 Présentation du projet **2**
1.1 Problématique . . . 2
1.2 Etude de l'existant . . . 2

2 Besoins fonctionnels **3**
2.1 Définitions . . . 3
2.2 Saisie des données : . . . 3
2.3 Préférences pour la saisie . . . 5
2.3.1 Liste des actions en accès direct . . . 5
2.3.2 Changement de possession automatique/manuel . . . 5
2.4 Gestion d'un tournoi . . . 5
2.4.1 Création d'un tournoi . . . 6
2.4.2 Ajout de statistiques et de résultats . . . 6
2.5 Import, Export des données . . . 6
2.5.1 Import . . . 6
2.5.2 Export . . . 7
2.5.3 Compatibilité entre les bases de données . . . 7
2.6 Visualisation des données . . . 7
2.6.1 Visualisation des statistiques après un match . . . 7
2.6.2 Visualisation des résultats d'un tournoi . . . 7
2.7 Besoins non fonctionnels . . . 7
2.7.1 Version minimale d'Android . . . 7
2.7.2 Résolution miniamle . . . 7

3 Architecture et conception **8**
3.1 Modèle de base de données . . . 8
3.2 Utilisation du pattern DAO (Data Access Object) . . . 9
3.3 Séquencement de la saisie d'une statistique . . . 10

Chapitre 1

Présentation du projet

1.1 Problématique

Au cours d'un match de basketball il est important pour chacune des équipes de connaître les statistiques des différents joueurs après un match, cela afin de pouvoir analyser la performance de chacun. Ces statistiques peuvent également être utile à l'entraîneur pendant le match. Lors d'un match professionnel, ces statistiques sont rentrées informatiquement par plusieurs statisticiens au bord du terrain. Mais pour un niveau amateur, cette tâche est souvent confiée à une seule personne. Cela pose plusieurs problèmes :

- Le statisticien étant seul, une saisie rapide des données est impérative pour éviter la perte d'informations. Un temps de saisie trop long signifie que le statisticien peut ne pas avoir le temps d'enregistrer toutes les statistiques.
- Généralement les remplacements ne se font pas de manière officielle (les joueurs rentrent et sortent sur le terrain sans en informer la table de marque). Le statisticien ne connaissant pas a priori les joueurs sur le terrain, il doit repérer lui-même le changement. Ce qui peut constituer une perte de temps conséquente.

1.2 Etude de l'existant

Plusieurs applications existent déjà mais sont difficiles d'utilisation dans un cadre amateur. En effet, nous avons constaté les défauts suivants :

- La saisie des données se fait généralement en plusieurs clics. On a donc une perte de temps qui va impliquer la perte de données.
- Possibilité de saisir des données uniquement pour les 5 joueurs présents sur le terrain. Ce qui implique que l'utilisateur doit repérer le changement qui a été effectué (ce qui n'est pas évident lorsque le statisticien ne connaît aucun joueur), puis prendre le temps de l'enregistrer sur la machine.
- Possibilité de saisir les statistiques pour seulement une équipe à la fois pour certaines applications. Ce qui oblige l'utilisateur à faire de nombreuses manipulations dans certains cas. Par exemple si on a un tir manqué de l'équipe qui attaque, puis le rebond d'un joueur de l'équipe qui défend. Alors l'utilisateur doit enregistrer le tir manqué, puis changer d'équipe, puis enregistrer le rebond défensif.

L'objectif est donc de fournir une application permettant une saisie rapide, et permettant de ne pas avoir à gérer les remplacements. L'application sera développée pour des tablettes Android. Afin d'étudier les statistiques après le match, l'application devra également permettre l'import, l'export et la visualisation des statistiques dans un format à déterminer.

Chapitre 2

Besoins fonctionnels

Dans cette partie nous allons définir les différentes fonctionnalités que devra fournir l'application afin de répondre aux attentes du client. Tout d'abord nous allons définir quelques notions nécessaires pour une bonne compréhension par la suite.

2.1 Définitions

Nous allons ici présenter différent termes qui seront employés au cours de ce document. En particulier nous allons expliquer brièvement certaines des actions que l'on peut rencontrer au cours d'un match de basket-ball.

Rapidité : On quantifie la rapidité d'une saisie en comptant le nombre de clics effectués par l'utilisateur :

1 clic = instantané

2 clics = rapide

3 clics et plus = lent

Clic : Action d'appuyer sur la tablette avec son doigt

Equipe offensive : Equipe en possession du ballon.

Equipe défensive : Equipe qui n'a pas le ballon.

Turnover `(To)` : Perte de balle attribuée à un attaquant.

Steal `(St)` : Vol de balle d'un défenseur. A noter qu'un `steal` d'un défenseur implique un `turnover` chez un attaquant. Mais qu'un `turnover` peut avoir lieu sans qu'il y ai eu de `steal`.

Rebound `(Rb)` : Récupération du ballon par un joueur d'une des deux équipes après une tentative de panier ayant échouée.

Assist `(As)` : Passe décisive.

Block `(B)` : Contre effectué par un défenseur sur un tir d'un attaquant.

2.2 Saisie des données :

L'application fournit une interface de saisie des données représentée dans la figure 2.1. L'interface se décompose en 6 modules.

Module 1 (Equipes A et B)

Description : Contient le nom des deux équipes jouant le match.
Fonctions :

- Enregistre l'équipe en possession du ballon via un clic sur le nom de l'équipe.
- Change les statistiques accessibles dans les modules 3 et 6 lors d'un changement de possession.

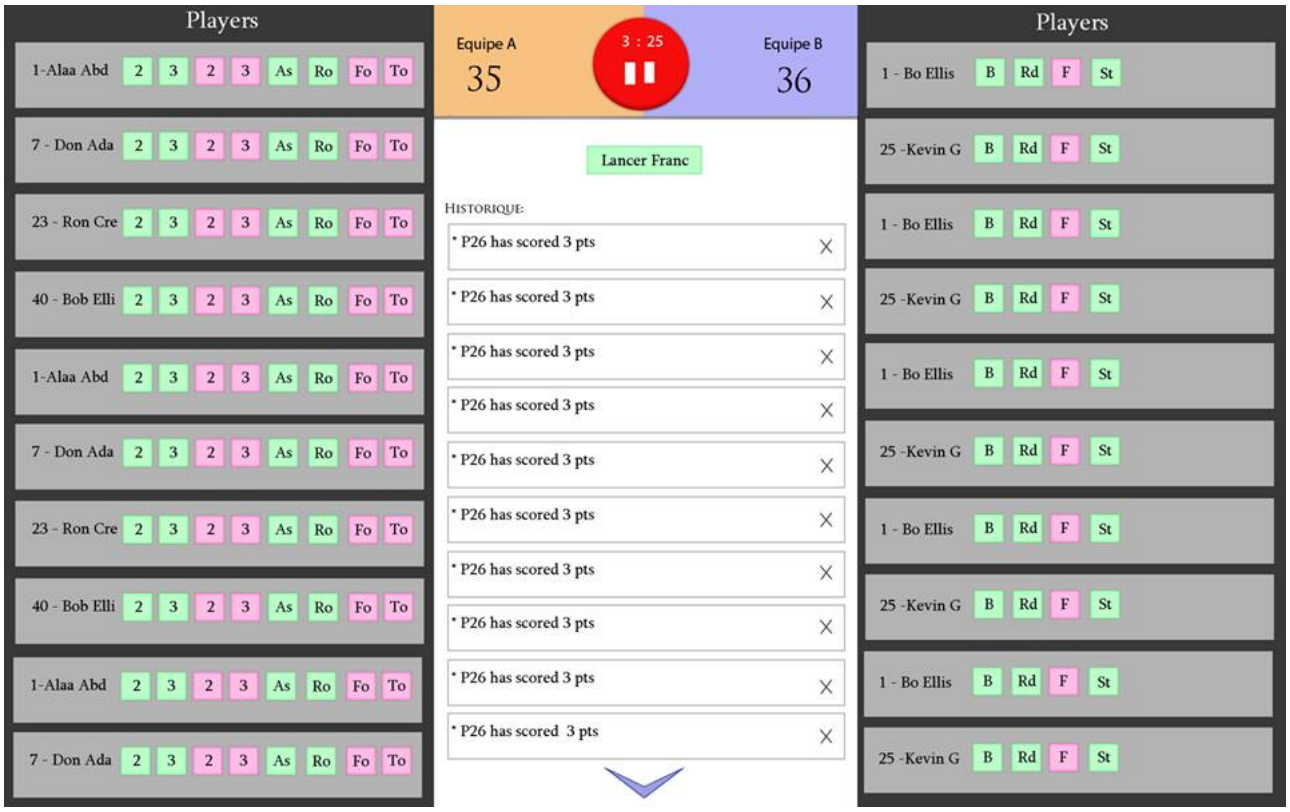


Figure 2.1 – Interface de saisie des statistiques

Module 2 (Tableau d'affichage)

Description : Affiche le score du match, le chronomètre, le numéro de quart-temps et contient un bouton start/stop.
Fonctions :

— Lance ou arrête le chronomètre en appuyant sur le bouton start/stop.

Module 3 et 6 (Modules de saisie instantanée)

Description : Contient les noms et numéros de tous les joueurs de l'équipe. Aucune distinction n'est faite entre les 5 joueurs de champ et les remplaçants. A côté de chaque joueur on dispose de plusieurs boutons représentant les statistiques considérées comme étant les plus fréquentes au cours d'un match. Compte tenu de la place disponible, le nombre maximum de boutons accessibles est fixé à 5. Les configurations possibles sont présentées dans la partie suivante.
Fonctions :

— Saisie instantanée des statistiques offensives/défensives pour tous les joueurs de l'équipe.
— Ajout automatique de la statistique dans l'historique
— Autorise la sélection d'un joueur pour le module 4.

Module 4 (Module de saisie lent)

Description : Contient une liste d'actions laissant du temps au statisticien pour enregistrer la statistique. Par exemple lors d'un lancer franc, le jeu est arrêté et le statisticien dispose de temps pour faire plusieurs clics. Ce module contient également les statistiques les moins utilisées. Cela inclu notamment la possibilité d'attribuer une faute à un joueur neutre pour chacune des deux équipes, dans le cas où une faute d'équipe est commise par exemple. Le choix des statistiques présentent dans ce module se fait au préalable (voir partie sur les Préférences pour la saisie). L'enregistrement de la statistique se fait en cliquant sur l'action, puis sur le joueur associé à l'action (2 clics).
Fonctions :

— Saisie dite "rapide" des statistiques

Module 5 (Historique)

Description : Recense toutes les statistiques saisies au cours du match et dans l'ordre chronologique. Le format d'affichage est `(Temps - Action - Joueur)`
Fonctions :
- Affichage chronologique des événements
- Suppression manuelle d'un événement
- Ajout manuelle d'un nouvel événement à un instant quelconque dans la chronologie du match.

2.3 Préférences pour la saisie

2.3.1 Liste des actions en accès direct

L'utilisateur peut configurer l'interface de saisie selon ses préférences. En effet selon les niveaux des joueurs sur le terrain, les statistiques les plus répandues diffèrent. Nous avons configurer 3 modes de saisies par défaut :

Amateur :
Pour un match de niveau amateur, le nombre de tir à 3 points réussis voir tentés au cours du match a tendance à être faible. Nous avons donc choisi la configuration suivante :
- Stats offensives : `2 pts réussi/manqué, RebondOff, Assist, Foul`
- Stats défensives : `Block, Steal, RebondDef, Foul`
- Stats accessibles 2-clics : `Lancer Franc, 3 pts réussi/manqué, Turnover`

Pro :
Pour un match de niveau professionnel, les tirs à 3 points deviennent fréquent. Comme une faute est synonyme d'arrêt de jeu, nous avons choisi de mettre cette statistique accessible en 2-clics pour les attaquants. De même une `assist` signifie qu'un panier a été marqué, le statisticien dispose donc d'un peu de temps vu qu'une remise en jeu est effectuée.
- Stats offensives : `2 pts réussi/manqué, 3 pts réussi/manqué, RebondOff`
- Stats défensives : `Block, Steal, RebondDef, Foul`
- Stats accessibles 2-clics : `Lancer Franc, Foul, Turnover, Assist`

Personnalisé :
Cette option permet à l'utilisateur choisir sa propre configuration pour saisir les statistiques. Plusieurs configuration personnalisées pourront être stockées dans l'application.

2.3.2 Changement de possession automatique/manuel

Nous avons défini dans le module 1 une fonction permettant à l'utilisateur d'indiquer quelle équipe est en possession du ballon. Par défaut cette action se fait manuellement. Cependant, le statisticien peut choisir un mode `automatique` qui change automatiquement la possession dans les cas suivants :
- `Steal` d'un joueur défensif. La possession revient alors à son équipe.
- `Rebound:` la possession est donnée au joueur ayant récupéré le ballon au rebond.
- `Foul:` le ballon est rendu à l'équipe ayant subi la faute.
- `Turnover:` l'attaquant perd le ballon et celui est récupéré par l'équipe adverse.
- `Tir à 2pts ou 3pts réussi:` le ballon est rendu à l'équipe ayant concédée le panier.

Le mode `automatique` implique que la saisie des statistiques doit se faire de manière ordonnée du fait que l'écran change en même temps que la possession. Ainsi après un panier à 2 points marqué, il faudra d'abord cliquer sur le joueur ayant fait l'`assist` puis sur le joueur ayant rentré le panier. Certains statistiques comme l'`assist` seront néanmoins disponibles en accès lent afin de rentrer la statistique dans le cas où le panier est enregistré en premier par l'utilisateur.

2.4 Gestion d'un tournoi

L'application devra également fournir une interface afin de créer et gérer un tournoi.

2.4.1 Création d'un tournoi

L'application devra gérer 2 types de tournois.
— Championnat : Tout le monde joue contre tout le monde.
— Tournoi : Phases de poules puis phases finales.

Les données du tournoi peuvent tout d'abord être ajoutées par l'intermédiaire de l'interface d'import des données depuis une source externe. Cependant, il faut qu'un ou plusieurs écrans de l'application permettent de créer un tournoi de toutes pièces. Il faut notamment pouvoir ajouter des équipes dans le tournoi, ainsi que de pouvoir modifier ces équipes et les joueurs qui les composent. Pour chaque équipe participant au tournoi, on veut aussi pouvoir sélectionner les joueurs de l'équipe participant effectivement au tournoi.

2.4.2 Ajout de statistiques et de résultats

Encore une fois, l'ajout de statistiques ou de résultats pour un match peut se faire par l'intermédiaire de l'interface d'import. L'ajout doit aussi pouvoir se faire directement dans l'application à travers une interface permettant de spécifier le score et les statistiques. Une option permettant de spécifier si des statistiques ont été prises pour le match en question doit aussi être présente afin de ne pas fausser d'autres statistiques globales par exemple.

2.5 Import, Export des données

L'application pourra communiquer avec un serveur via la connexion wifi, afin d'exporter ou d'importer des données sur le serveur. Les réglages se font via une page dédiée sur l'application. La transmission des données se fait dans un format XML.

2.5.1 Import

Le format XML étant celui choisi, nous devons d'abord définir les schémas conforment à la base de données que nous allons utiliser, ce en fonction de ce que l'on veut importer (ou exporter).

— Import d'une équipe.
  Le schema nous donne :

```
<equipe id = "", couleur="", nom="">
  <photo>
        Le lien vers la photo ou la photo elle meme en fonction de notre
           choix.
  </photo>
</equipe>
```

  Nous pouvons donc à partir de ce dernier recupérer les informations de la table équipe de la base de données.

— Import d'un joueur. Il est similaire à celui de l'équipe.

```
<joueur id="", nom="", prenom="", pseudo="", naissance="", sexe="", nom\
   _utilisateur="", md5\_mdp="">
  <photo> </photo>
</joueur>
```

— Import des résultats (classements) d'un tournoi.
  Le schema XML correspondant aux résultats d'un tournoi est alors similaire à la table de classement tournoi.

```
<resultournoi id="", place="", regle_equipe="">
  <equipe> Ici tout ce que contient une equipe </equipe>
  <tournoi> Toutes les informations relatives aux tournoi </tournoi>
</resultournoi>
```

Le reste des imports est basé sur le même principe. C'est à dire que les elements propre à la table sont des attributs et ceux qui sont référencés ailleurs sont des noeuds fils.

Sur le même principe on peut également avoir les différents imports suivants :
— Import des statistiques d'un joueur (sur un match, sur un tournoi).
— Import des équipes et des matchs prévus pour un tournoi.
— Import des résultats d'un match, avec ou sans statistiques.

Les données importées seront intégrées dans la base de données locale à l'application.
Elles peuvent être stockées dans la mémoire de la tablette ou sur la carte SD.

2.5.2 Export

L'export se fait aussi en utilisant les mêmes schémas. Le principe est exactement le même car importer et exporter n'est autre que recevoir ou envoyer des données de la base locale vers le serveur. D'autre part, on pourra aussi exporter des résultats d'un match, d'un tournoi, les statistiques d'un joueur ou d'une équipe au format PDF pour plus de lisibilité lors d'une impression ou autre. Ainsi donc l'export se résume en deux parties,

- Envoyer des données de la base locale vers le server au format XML.
- Rendre les resultats et les statistiques au format PDF pour une impression ou une meilleure lisibilité. Les données dans ce cas seront prises soit depuis le serveur, soit depuis la base locale.

2.5.3 Compatibilité entre les bases de données

La base présente sur le serveur sera une base de type MySQL, et celle sur le support android sera en SQLite. Mais de par le fait que le schema de la base de données reste le même, nous auront deux DAO différents. Celui qui se sert d'écrire dans la base SQLite et l'autre dans la base MySQL. Ces deux DAO prendront en entrée du XML et écriront directement dans la base correspondante. Ils se chargeront aussi de regler les différents problèmes qui pourraient survenir, à savoir des erreurs sql, des données déjà présentes dans la base de données, voir d'éventuels conflits.

2.6 Visualisation des données

Après qu'un match ou qu'un tournoi ait eu lieu, l'utilisateur veut pouvoir visualiser l'ensemble des statistiques saisies. Il existe plusieurs types de visualisations possiblent que nous allons présenter dans cette partie.

2.6.1 Visualisation des statistiques après un match

Au cours du match ou à la fin. L'application proposera :

- Une vue d'ensemble du match avec l'affichage du score et les statistiques principales des deux équipes.
- Une visualisation des statistiques détaillées du match pour chaque joueur.

2.6.2 Visualisation des résultats d'un tournoi

On veut également pouvoir visualiser les résultats du tournoi en cours ou bien d'un tournoi joué précédemment. L'application propose donc :

- Une vue d'ensemble des tournois présents dans la base de données de la tablette.
- Une vue d'ensemble du calendrier des matchs joués et à venir pour le tournoi sélectionner.
- Pour chaque poule, un affichage avec le classement de la poule et le calendrier des matchs pour la poule. On pourra également afficher le tableau des phases finales du tournoi.
- Une visualisation des résultats de l'équipe sélectionnée sur un tournoi.
- Classements des meilleurs joueurs sur le tournoi.
- Statistiques de chaque joueur sur le tournoi.

2.7 Besoins non fonctionnels

2.7.1 Version minimale d'Android

Après concertation la version minimale d'Android ciblée pour développer cette application pour tablette est la version 3.0 (Honeycomb). Cette version a été choisi car elle permet de développer l'application pour à peu près toutes les tablettes Android existantes.

2.7.2 Résolution miniamle

Concernant la résolution minimale, nous avons choisi de nous adapter à une résolution standard de 1280x800, ce qui correspond à une tablette 10 pouces.

Chapitre 3

Architecture et conception

3.1 Modèle de base de données

La base de donnée est fortement inspirée d'un modèle pré-existant proposé par le client. Comprenant à la fois les saisies liées aux actions se déroulant pendant un match et les entités relatives à la structure de l'équipe et du tournoi.

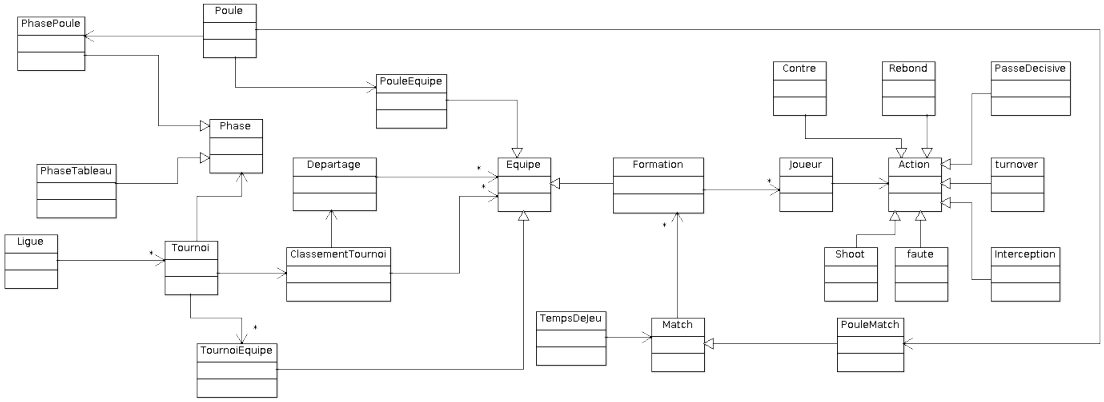


FIGURE 3.1 – Modèle du Projet

La figure précédente nous permet de bien identifier la séparation entre les objets liés à la structure du match (partie gauche) et les objets concernant la saisie de données (partie droite).

Structure du match

Un match a lieu au cours d'un tournoi. Nous avons toute une structure dans notre modèle permettant de gérer le tournoi du point de vue de la base de données. Ainsi :

- Un tournoi prend place au sein d'une ligue. On enregistre les équipes prenant part au tournoi.
- Le déroulement d'un tournoi se divise en une phase de poule et une phase finale appellée phase tableau.
- Pour chaque match, on défini la sélection des joueurs de l'équipe participant au match dans la table `Formation`.
- A la fin du tournoi, on établit un classement final où sont départagées les équipes.

Saisie des données

La saise des données est assez simple, le principe étant que pour chaque joueur de la formation choisie pour le match, l'on peut enregistrer une nouvelle action à n'importe quel moment du temps de jeu. L'ensemble des actions permet d'enregistrer n'importe quelle statistique. A noter qu'il existera pour chacune des formations un joueur neutre permettant d'attribuer des fautes d'équipe.

3.2 Utilisation du pattern DAO (Data Access Object)

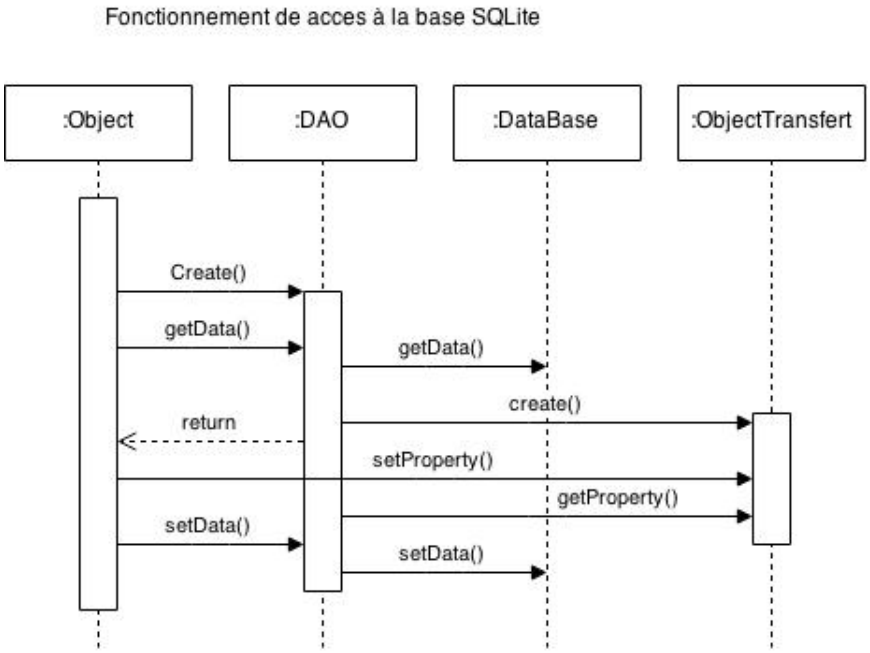


FIGURE 3.2 – Diagramme séquence DAO

Une classe DAO peut être associé à chaque classe Object (dans notre cas des classes qui implémentent actions) et ainsi être unique pour chaque type d'actions.

L'objet de Transfert permet quant à lui de véhiculer dans ses attributs les données transferées, cet objet ne possède que des accesseur et mutateur.

Pourquoi utiliser un DAO, l'objet ne connait pas l'existence du type de stockage employé et ce dernier peut être changer ou évoluer facilement.

L'objet de transfert :

L'objet de transfert permet de garder temporairement des informations dans la mémoire de la machine avant des les insérer dans le support de stockage (Base de données par exemple). Cette objet se montre utile lorsque plusieurs requêtes doivent être envoyées vers le support de stockage. Ainsi information pas encore écrite dans la base de données ou lu par notre application par exemple seront stockées dans la mémoire de la machine pour y être traité comme ils doivent l'être. Néanmoins, malgré que cette objet n'est que des accesseur et mutateurs, il peut être intéressant d'enfiler les objets de transferts dans des fifo par exemple pour que les données soient traitées dans un ordre chronologique.

3.3 Séquencement de la saisie d'une statistique

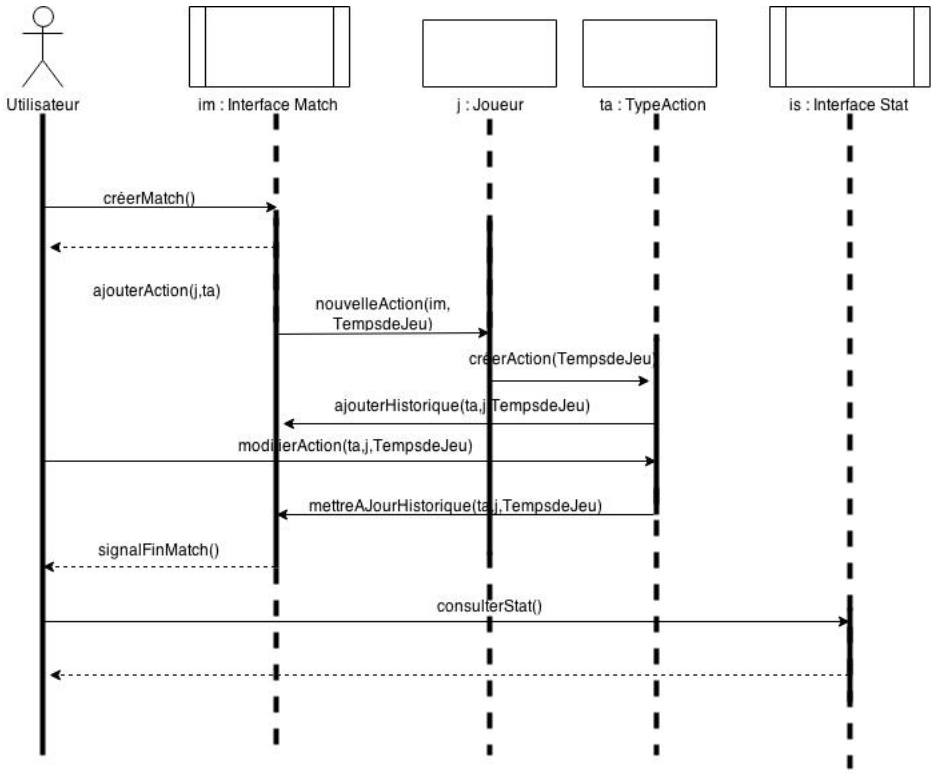


FIGURE 3.3 – Diagramme de Séquence

La figure précédente présente un diagramme de séquence qui montre ce qui se passe au niveau de l'application lorsque l'utilisateur utilise l'interface de saisie.

Lors de la création d'une nouvelle action, un objet métier `action` est créé par le code de l'interface suite au clic de l'utilsateur. Alors l'objet métier est traité par le module de DAO qui va permettre de stocker la statistique dans la base de données.

FIGURE 3.4 – Modèle DAO

BASTATS

B. Manuel d'utilisation

BASTATS

Saisie de statistiques en temps réel - Manuel d'Utilisation

Sabir Reda, Carrié Mathieu, Zouad Lotfi
Boudjeltia Retda, Barré Jean, Adotevi Lionel

Chabot Romain

PROJET DE FIN DE DEUXIÈME ANNÉE, ENSEIRB-MATMECA

Projet réalisé sous la supervision de M. Yves Métivier dans le cadre d'un Master en Informatique à l'ENSEIRB-MATMECA, Bordeaux (France).

30 Mars 2015

BASTATS

Table des matières

1 Configuration système et installation 5

1.1 Configuration minimum 5

1.2 Installation 5

1.3 Lancement de l'application 6

2 Préférences utilisateur 7

3 Lancement d'un match 9

3.1 Sélection des équipes 9

3.2 Sélection des joueurs 10

4 Interface de match 13

4.1 Table des statistiques 13

4.2 Équipes 14

4.3 Actions 15

4.4 Historique 15

4.5 Chronomètre 16

4.6 Gestion de la possession 16

4.7 Visualisation des statistiques 16

4.8 Mise en garde 18

5 Mode tournoi 19
5.1 Création d'un nouveau tournoi 19
5.2 Menu du tournoi 21
5.2.1 Classement du tournoi 21
5.3 Création d'une nouvelle phase 22
5.3.1 Phase de poule 22
5.3.2 Phase tableau 24
5.4 Phase de poule 24
5.5 Phase finale 25
5.6 Import du résultat match 27

6 Import et export 29
6.1 Import d'un équipe 29
6.2 Export complet d'un match 31
6.3 Génération d'un PDF 33

1. Configuration système et installation

1.1 Configuration minimum

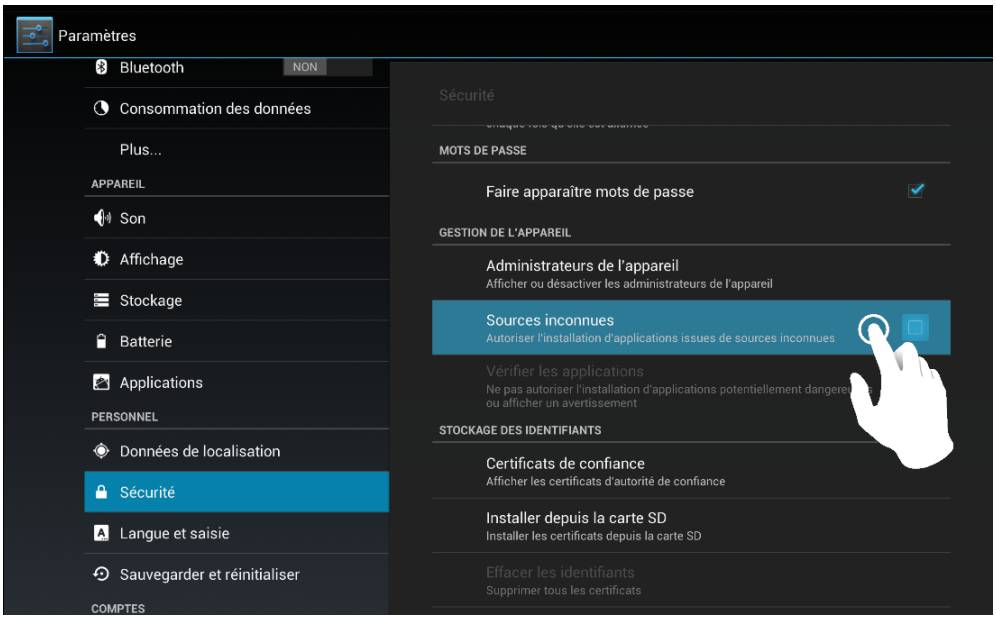
L'application est développée pour tablette Android.

VERSION MINIMUM: 3.0 (HONEYCOMB)
RÉSOLUTION OPTIMALE: 1280x800

Vous pouvez vérifier les paramètres systèmes de votre tablette en allant dans
Paramètres -> À propos du téléphone

1.2 Installation

L'application n'étant pas signée ni présente sur le Google Play Store, il est nécessaire d'autoriser l'installation d'applications provenant de sources inconnues. Pour celà, vous devez aller dans **Réglages -> Sécurité** puis activé l'option **Sources inconnues**. Cette option pourra être de nouveau désactivée après l'installation.



Désormais il vous suffit de trouver le fichier `Bastats.apk` sur votre tablette et de cliquer dessus. L'installation se fait automatiquement.

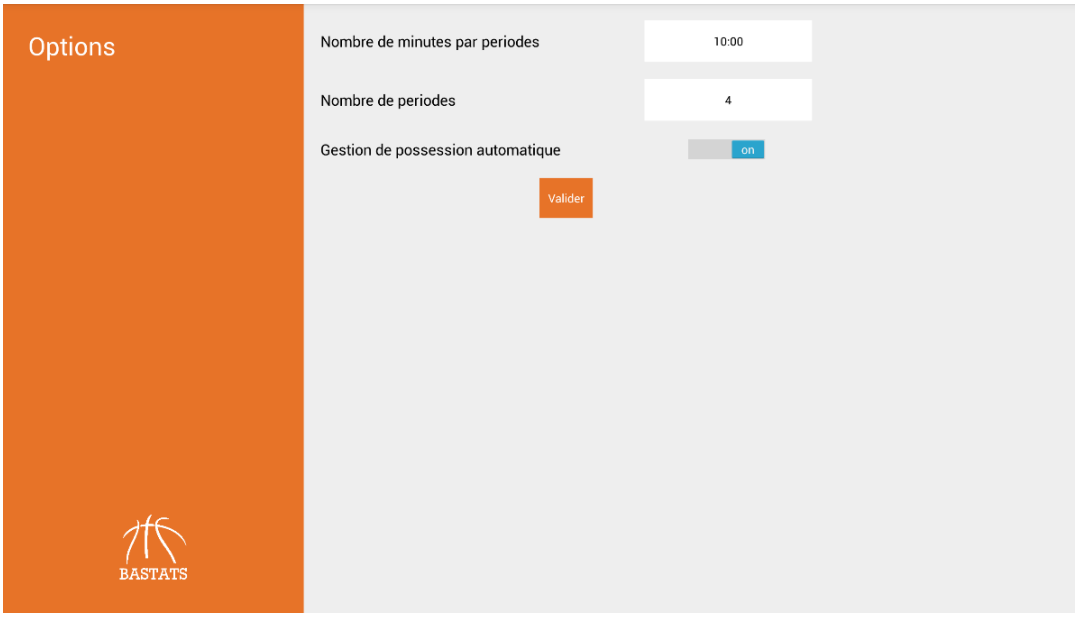
1.3 Lancement de l'application

Pour lancer le jeu, cliquez sur l'icône **BASTATS**

. L'application se lance en affichant le menu principal. Le menu principal permet de lancer un nouveau match, lancer le mode tournoi et définir les préférences de l'application.

2. Préférences utilisateur

Les préférences utilisateur peuvent être définies en cliquant sur le bouton **OPTIONS**.



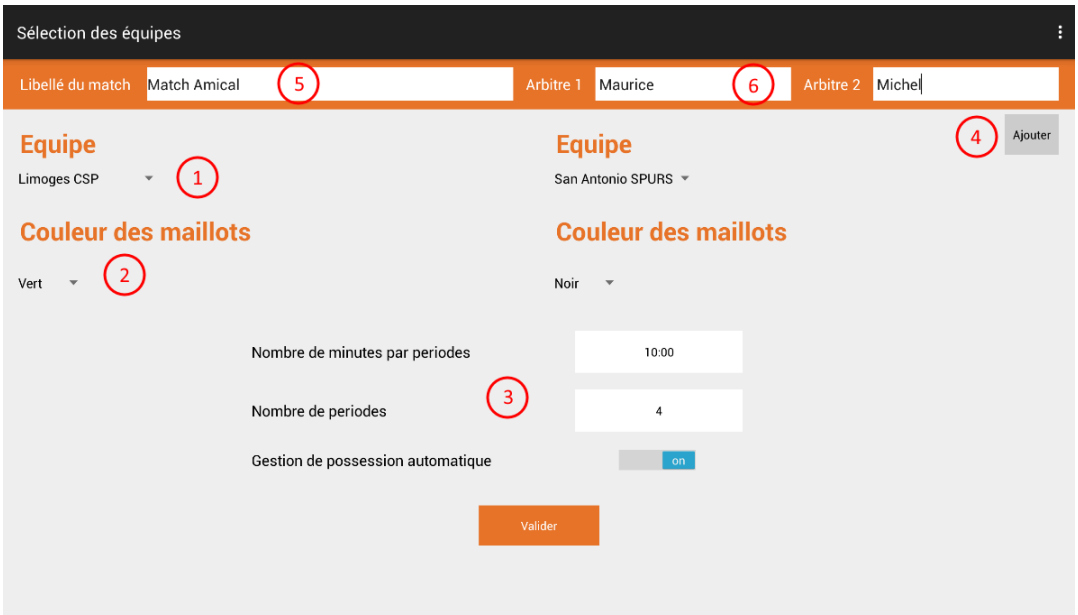
Durée période: Permet de définir le nombre de minutes que dure une période.
Nombre de période: Défini le nombre de périodes à jouer pour le match.
Gestion de possession automatique: Cette option permet de gérer automatiquement les changements de possession au cours d'un match. Les actions entraînants un changement de possession sont détaillées dans la partie 4.6.

3. Lancement d'un match

Un match peut être démarré soit depuis l'écran d'accueil, soit depuis le menu de navigation d'un tournoi.

3.1 Sélection des équipes

L'écran suivant est affiché à chaque lancement de match.



1) Choix des équipes: Dans le cas où le match est lancé depuis l'écran d'accueil, l'utilisateur peut sélectionner les deux équipes qu'ils souhaitent confronter. Si le match est lancé depuis un tournoi, le choix des équipes est défini au préalable et ne peut être modifier.

2) Choix des couleurs: La couleur de l'équipe est proposée par défaut mais celle-ci peut être modifié pour un match.

3) Options du match: Il est possible de modifier le nombre de périodes et leur durée pour le match à venir. La gestion de possession peut aussi être modifiée de manière ponctuelle.

4) Ajouter une équipe: Ce bouton amène vers une interface permettant d'intégrer une nouvelle équipe à votre base de données. Son fonctionnement est détaillé dans la partie TODO.

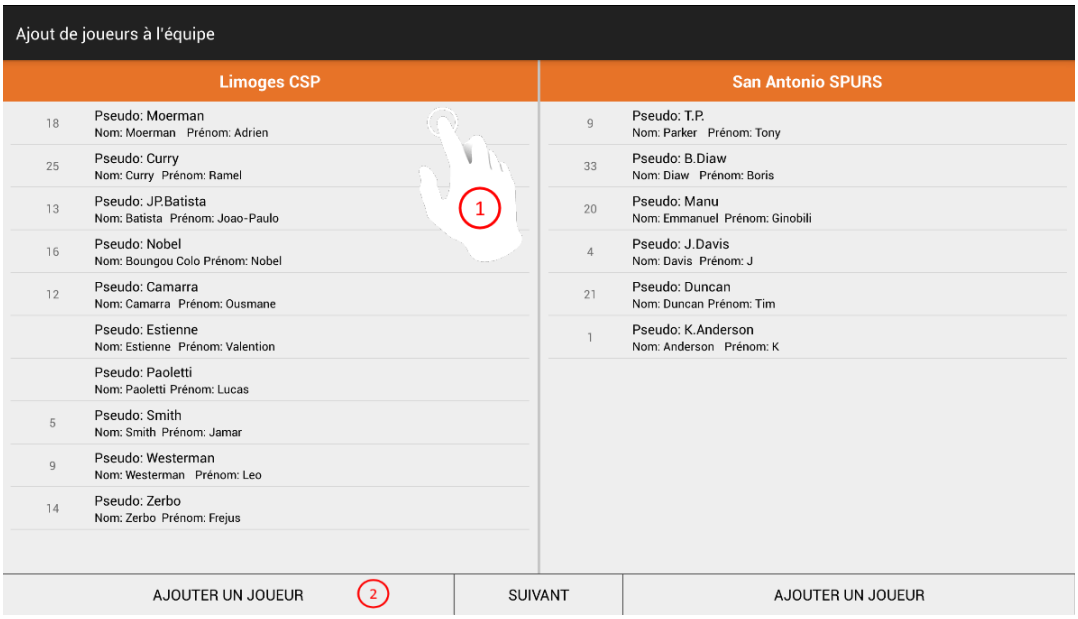
5) Libellé: Libellé du match. Celui-ci peut toujours être modifié.

6) Nom des deux arbitres

3.2 Sélection des joueurs

La sélection des joueurs pour le match se fait en deux temps.

Liste des joueurs disponibles



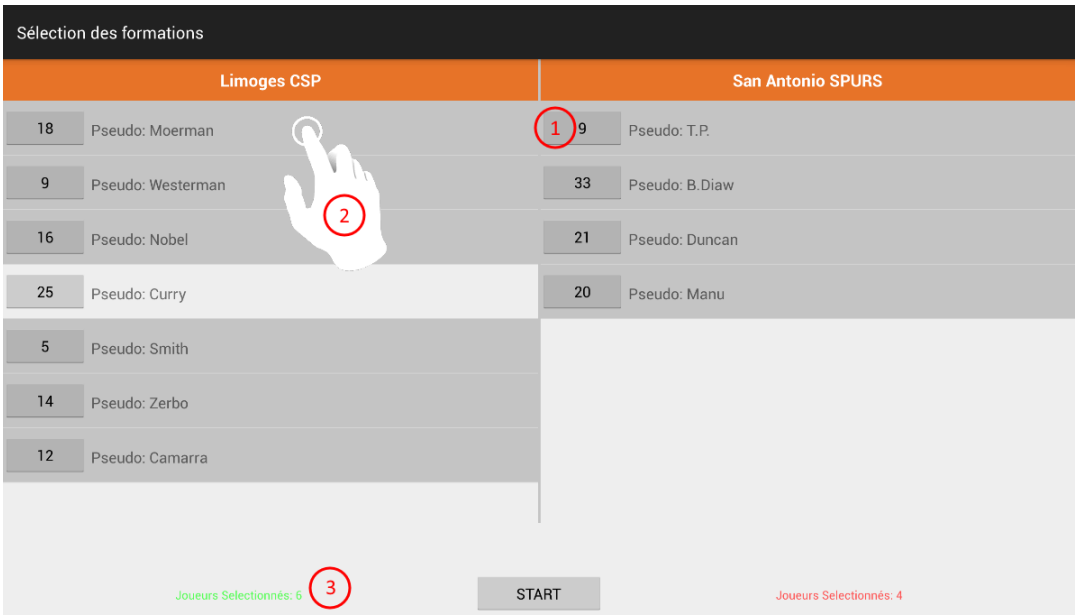
1) Suppression d'un joueur: Le joueur sélectionné est déclaré comme ne faisant plus partie de la liste des joueurs de l'équipe. Cependant le joueur n'est pas supprimé de la base de données.

2) Ajout d'un nouveau joueur: Permet d'ajouter un nouveau joueur à l'équipe en indiquant les informations **nom, prénom et pseudo**. Il est aussi possible d'ajouter un joueur déjà existant dans la

base de données à l'équipe choisie. L'appartenance d'un joueur à une équipe n'est pas exclusif et un même joueur peut se retrouver deux fois dans la même équipe.

Sélection des joueurs

Une fois la liste des joueurs bien définie, il est possible de les sélectionner ou non pour le match à venir.



1) Choix du numéro: Un clic sur ce bouton permet de définir le numéro de maillot du joueur. Ce numéro sera gardé comme numéro par défaut.

2) Joueur sélectionné: À noter qu'un joueur sélectionné pour un match sera sélectionné par défaut pour le match suivant de la même équipe.

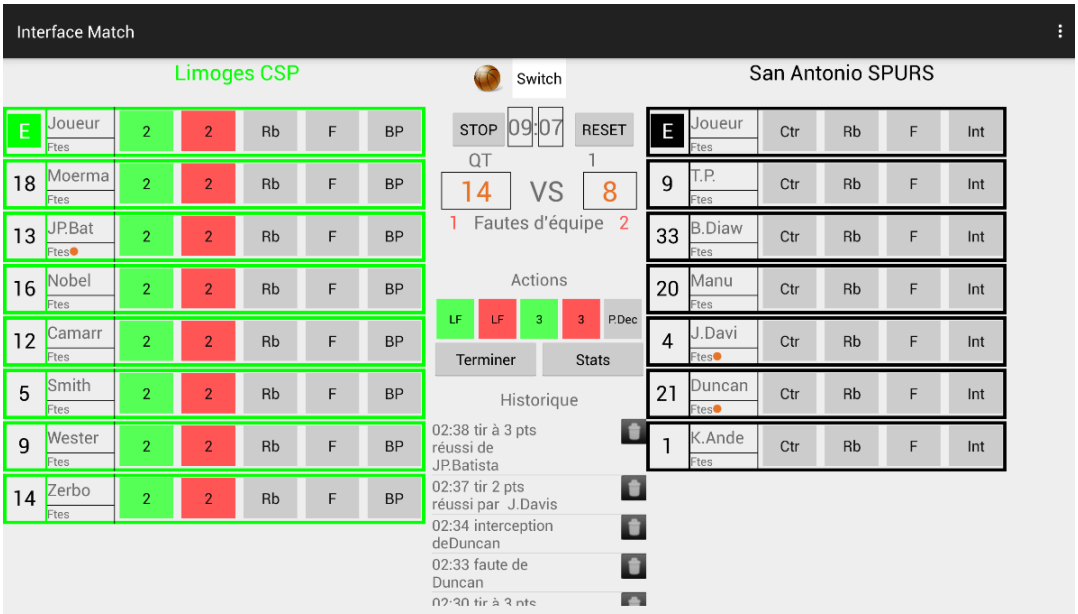
3) Nombre de joueurs: Le nombre minimum de joueurs est de 5 pour un match de basket. Cependant il est possible de lancer un match avec aucun joueur sélectionné pour le match.

JOUEUR ÉQUIPE: Le JOUEUR ÉQUIPE est défini automatiquement et permet d'enregistrer une statistique qui sera attribuée à l'équipe, plutôt qu'à un joueur particulier. Le joueur équipe est présent à chaque match et ne peut pas être retiré.

BASTATS

4. Interface de match

Une fois la sélection des joueurs pour chaque équipe établie, le match est lancé et l'enregistrement des statistiques se fait via l'écran suivant :



4.1 Table des statistiques

Chaque bouton correspond à une statistique précise. Nous allons détailler la signification de chacun des boutons ainsi que leur utilisation pour les cas particuliers.

2 : Tir à 2 points réussi.

2 : Tir à 2 points manqué.

3 : Tir à 3 points réussi.

3 : Tir à 3 points manqué.

LF : Lancer franc réussi.

LF : Lancer franc manqué.

BP : Ballon perdu par un joueur de l'équipe offensive.

Int : Ballon volé par un joueur de l'équipe défensive.

Ctr : Joueur contrant un tir adverse.

Rb : Rebond pris par le joueur après un tir manqué.

F : Faute commise par un joueur. Un clic rapide sur le bouton **FAUTE** enregistre la faute comme étant une faute personnelle. Mais il est également possible laisser le doigt appuyé sur le bouton afin de préciser si la faute est technique, antisportive ou disqualifiante grâce à une boîte de dialogue.

4.2 Équipes

L'équipe offensive est l'équipe étant en possession du ballon et est indiquée par l'icône . L'équipe n'ayant pas le ballon est l'équipe défensive.

1) Numéro du joueur: Le numéro du joueur pouvant être une chaîne de caractère celui-ci est tronqué à 2 caractères.

2) Pseudo du joueur: Un pseudo pouvant être tronquée est affiché.

3) Ftes: Nombre de fautes commises par le joueur sur l'intégralité du match. L'affichage s'arrête à 5 fautes.

4) Boutons de saisie rapide: Un clic sur chacun de ses boutons permet d'enregistrer directement l'action et de l'attribuer au joueur en question. Une exception est faite pour le bouton faute, dont

l'utilisation est détaillée dans la section Table des statistiques.

Joueur équipe

Le joueur équipe vous permet d'enregistrer une statistique et d'attribuer celle-ci à l'équipe plutôt qu'à un joueur en particulier. Cela peut être utile notamment lorsque le joueur ayant effectué l'action n'a pas été bien identifié.

4.3 Actions

Cette partie de l'écran contient les statistiques qui peuvent être enregistrées en procédant à deux clics. La façon de procéder est la suivante :

1. Sélection de la statistiques à enregistrer.
2. Sélection du joueur à qui est attribuée la statistique.

Conseil : Entre le premier et le deuxième clic, il est impossible d'annuler le premier clic. Ainsi en cas d'erreur sur le premier clic, il est possible d'annuler en enregistrant une action au hasard, puis en la supprimant dans l'historique.

4.4 Historique

L'historique affiche dans l'ordre chronologique la liste des statistiques enregistrées pour le match. Il est possible de supprimer une saisie en cliquant sur l'icône

4.5 Chronomètre

Start: Permet de démarrer le chronomètre.

Chronomètre: Un clic sur le chronomètre permet à l'utilisateur d'initialiser le temps comme il souhaite.

Reset: Remet le chronomètre à zéro.

Nombre de périodes: Le nombre de période (QT) peut être incrémenté en cliquant sur le numéro affiché.

Score des équipes: Le score évolue dynamiquement au cours du match.

Nombre de fautes: Le nombre de fautes par équipe pour la période en cours.

4.6 Gestion de la possession

Le bouton **SWITCH**, situé en haut de l'écran, permet de changer manuellement la possession de balle sur l'écran. Comme évoqué précédemment la gestion de la possession peut se faire automatiquement. Lorsque l'option est activée un changement de possession dans les cas suivants :

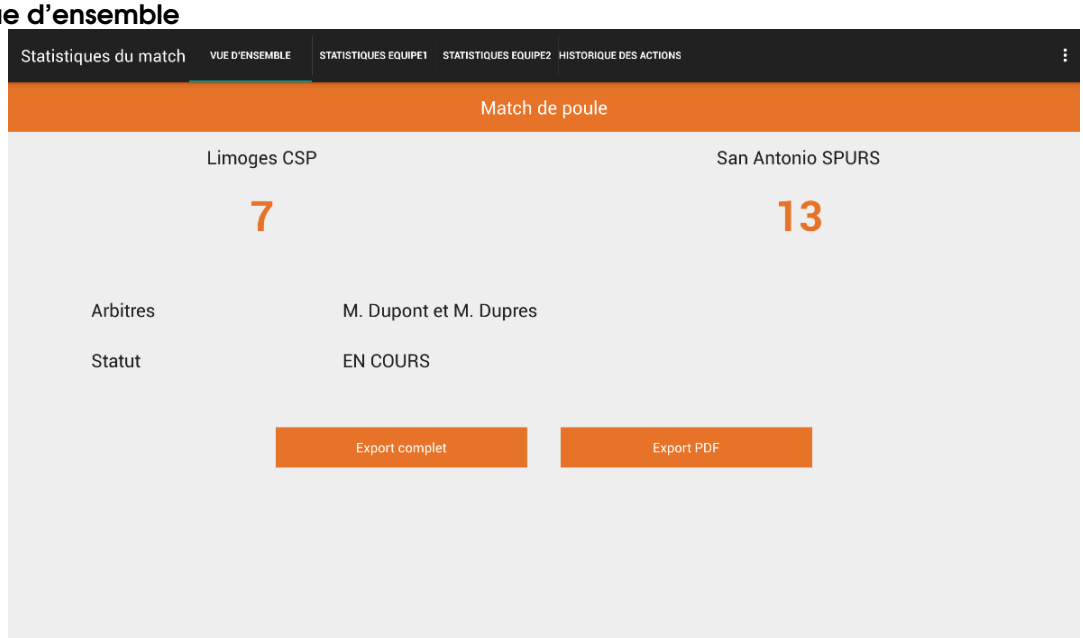
- Tir à deux points réussi
- Tir à trois points réussi (ne fonctionne pas malheureusement)
- Interception de la défense
- Rebond pris par la défense
- Perte de balle de l'attaque
- Faut offensive

Conseil: Lorsque le changement de possession est automatique veillez à saisir le cas échéant la passe décisive avant le tir réussi afin de gagner du temps.

4.7 Visualisation des statistiques

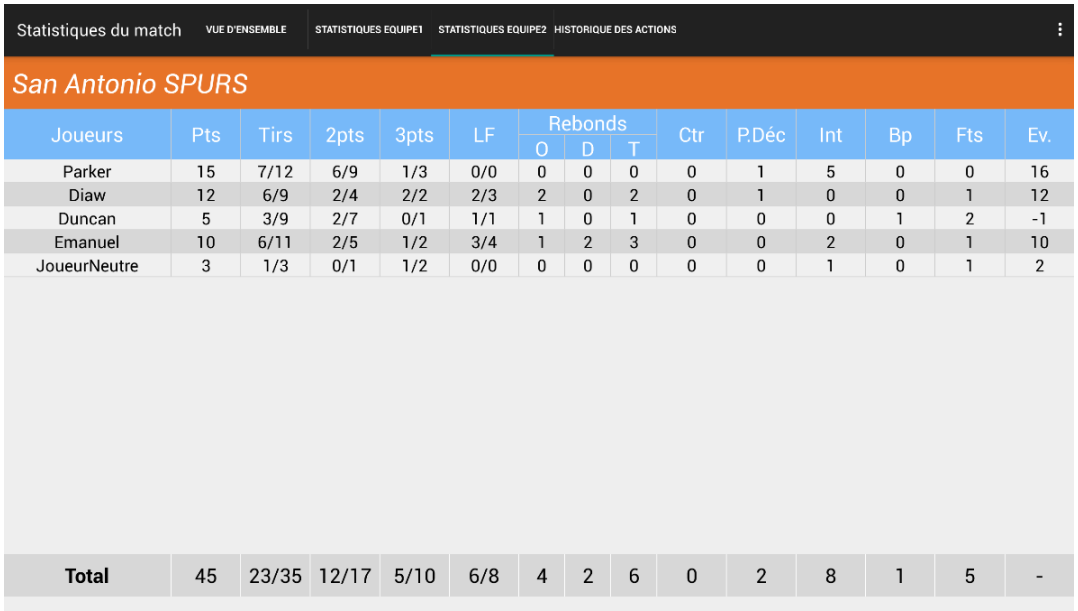
L'interface match a un bouton **STATS** permet de lancer un écran de visualisation des statistiques. Les statistiques sont accessibles à tout moment du match. Cet écran se divise en 4 parties, représentées par 4 onglets en haut de l'écran.

Vue d'ensemble



La vue d'ensemble contient les informations principales à propos du match. C'est à partir de cet écran que vous pouvez générer un fichier `.pdf` du match ou exporter les données.

Statistiques

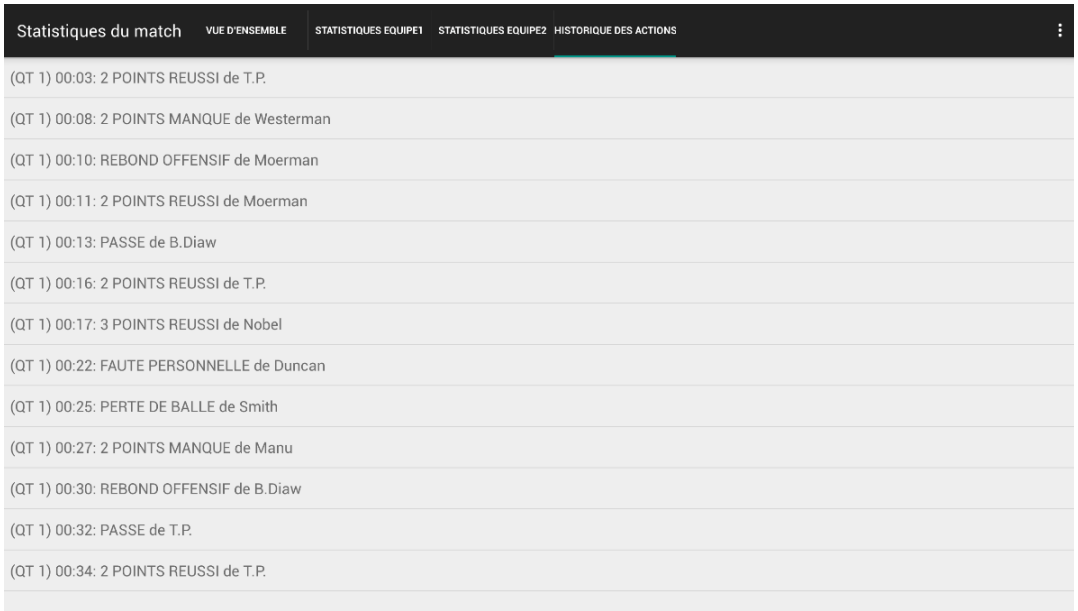


Joueurs	Pts	Tirs	2pts	3pts	LF	Rebonds			Ctr	P.Déc	Int	Bp	Fts	Ev.
						O	D	T						
Parker	15	7/12	6/9	1/3	0/0	0	0	0	0	1	5	0	0	16
Diaw	12	6/9	2/4	2/2	2/3	2	0	2	0	1	0	0	1	12
Duncan	5	3/9	2/7	0/1	1/1	1	0	1	0	0	0	1	2	-1
Emanuel	10	6/11	2/5	1/2	3/4	1	2	3	0	0	2	0	1	10
JoueurNeutre	3	1/3	0/1	1/2	0/0	0	0	0	0	0	1	0	1	2
Total	45	23/35	12/17	5/10	6/8	4	2	6	0	2	8	1	5	-

Pts: Nombre de points marqués par le joueur.
Tirs: Rapport nombre de tirs réussis sur nombre de tirs tentés.
2pts: Rapport nombre de tirs à deux points réussis sur nombre de tirs à deux points tentés.
3pts: Rapport nombre de tirs à deux points réussis sur nombre de tirs à deux points tentés.
LF: Rapport nombre de lancer francs réussis sur nombre de lancer francs tentés.
Rebonds O: Nombre de rebonds offensifs.

Rebonds D: Nombre de rebonds défensifs.
Rebonds T: Nombre total de rebonds.
Ctr: Nombre de contre réussis.
P.Déc: Nombre de passes décisives réussies.
Int: Nombre d'interceptions effectuées.
Bp: Nombre de balles perdues.
Fts: Nombre de fautes commises.
Fts: Nombre de fautes commises.
Év.: Évaluation du joueur sur le match calculée selon la formule :
$\text{ÉVALUATION} = \text{PTS} + \text{RB}_{tot} + \text{P.DÉC} + \text{INT} + \text{CTR} + (\text{TIRS}_{\text{RÉUSSIS}} - \text{TIRS}_{tents}) + (\text{LF}_{russis} - \text{LF}_{tents}) - \text{BP}$.

Historique



L'historique permet de consulter la liste des actions ayant été enregistré au cours du match.

4.8 Mise en garde

Attention lorsque vous jouez un match, il est fortement déconseillé de quitter le match alors qu'une saisie des statistiques a été démarré. Si cela se produit les données enregistrées restent dans la base de données mais ne sont remise à jour lorsque le match est lancé de nouveau.

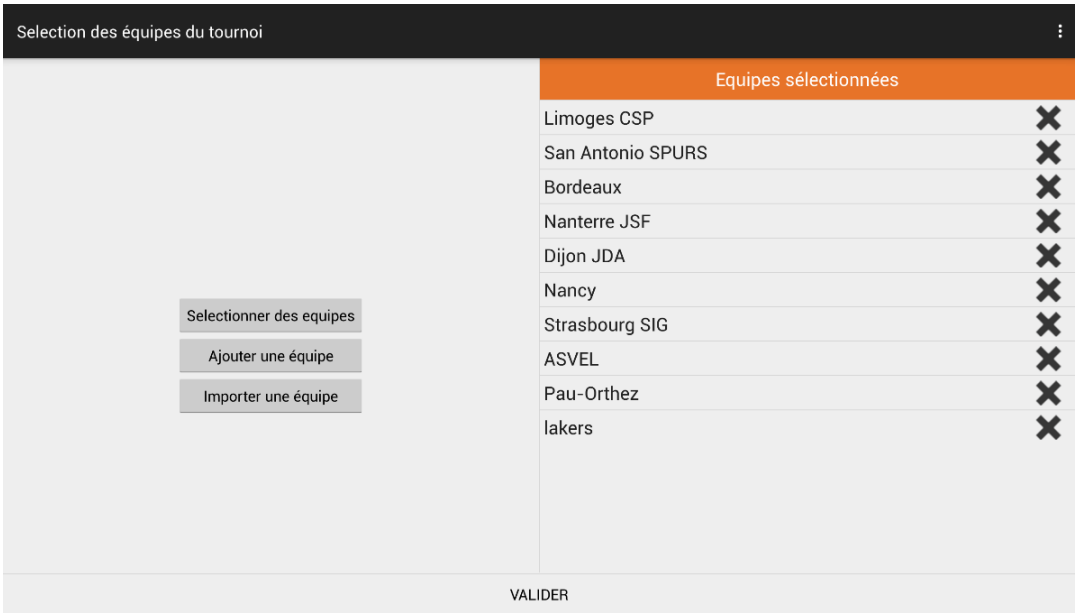
5. Mode tournoi

Le mode tournoi vous permet de créer et d'organiser intégralement votre propre tournoi. Il se lance en cliquant sur le bouton **TOURNOI** de l'écran d'accueil. Depuis l'accueil du mode tournoi, il est possible de supprimer un tournoi existant et tous les matchs liés à ce tournoi, de reprendre un tournoi en cours ou de créer un nouveau tournoi.

5.1 Création d'un nouveau tournoi

Le **nom** et le **lieu** du tournoi son requis afin de créer un nouveau tournoi. Ensuite il est demandé d'effectuer une sélection des équipes participants au tournoi.

Sélection des équipes

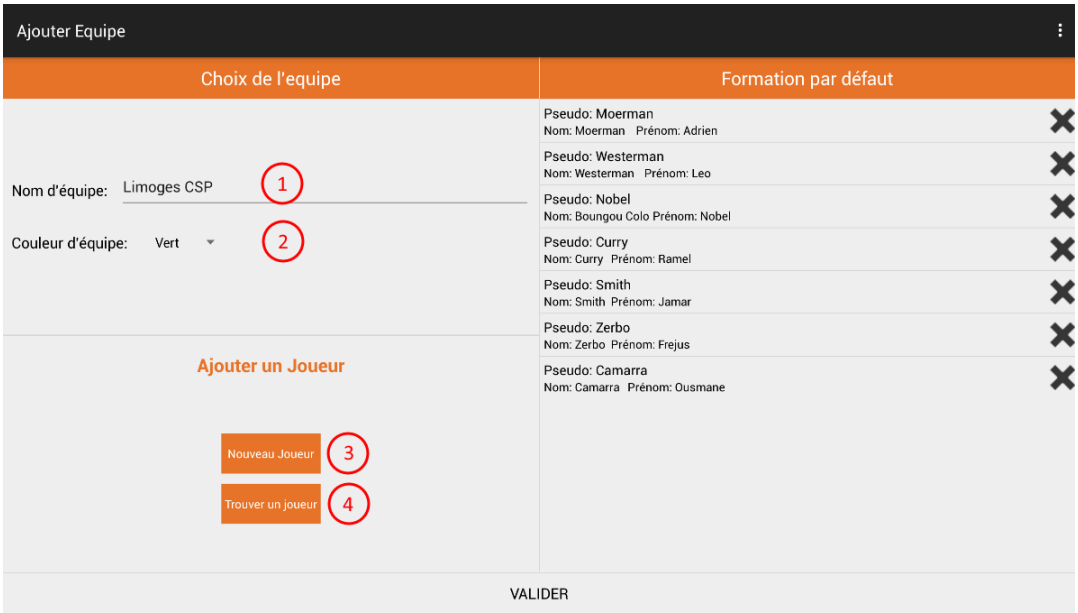


Vous pouvez sélectionner autant d'équipes que souhaité parmi la liste des équipes présentent dans la base de données. Cette liste est visible en cliquant sur le bouton **SÉLECTIONNER DES ÉQUIPES**. Les équipes sélectionnées sont affichées sur la partie droite de l'écran. Il est possible de supprimer une équipe de la sélection en cliquant sur l'icône ✖.

Import d'équipes

Il est possible d'importer des équipes en cliquant sur le bouton **IMPORTER UNE ÉQUIPE**. Afin d'importer une équipe vérifier au préalable qu'il existe un moins un fichier dans le répertoire `Android/data/BastatsFiles`. Sinon il est possible que l'application s'arrête.

Ajout d'une nouvelle équipe



1) Nom de l'équipe

2) Couleur de l'équipe: Attention la couleur sélectionnée ici sera la couleur par défaut **définitive** de l'équipe.

3) Nouveau joueur: Permet d'ajouter un nouveau joueur à l'équipe.

4) Trouver joueur: Permet de trouver un joueur existant dans la base de données et de le lier à l'équipe.

5.2 Menu du tournoi

Menu déroulant: Le menu déroulant de gauche peut être rendu visible ou invisible en cliquant sur l'icône ≡ située en haut à gauche de l'écran.

Vue d'ensemble: Par défaut la vue d'ensemble est affichée lorsque le menu est caché. Cette vue d'ensemble permet en particulier de modifier le classement final du tournoi.

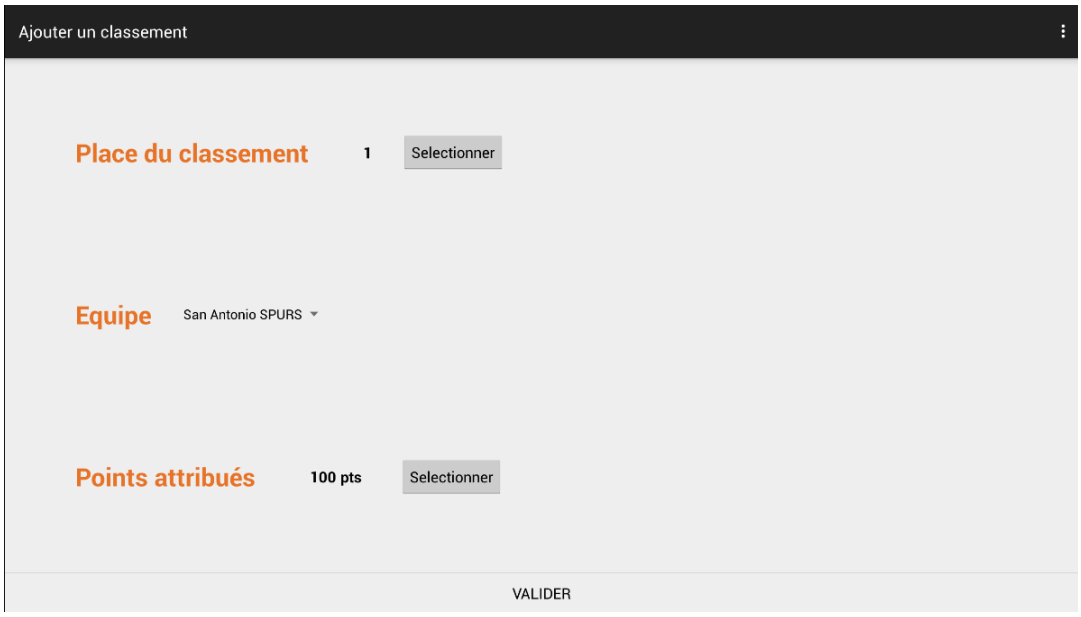
Statistiques: Lance l'écran de statistiques du tournoi.

Nouvelle phase: Permet de créer un nouvelle phase pour le tournoi.

Liste des phases: La phase sélectionnée est affichée à l'écran.

5.2.1 Classement du tournoi

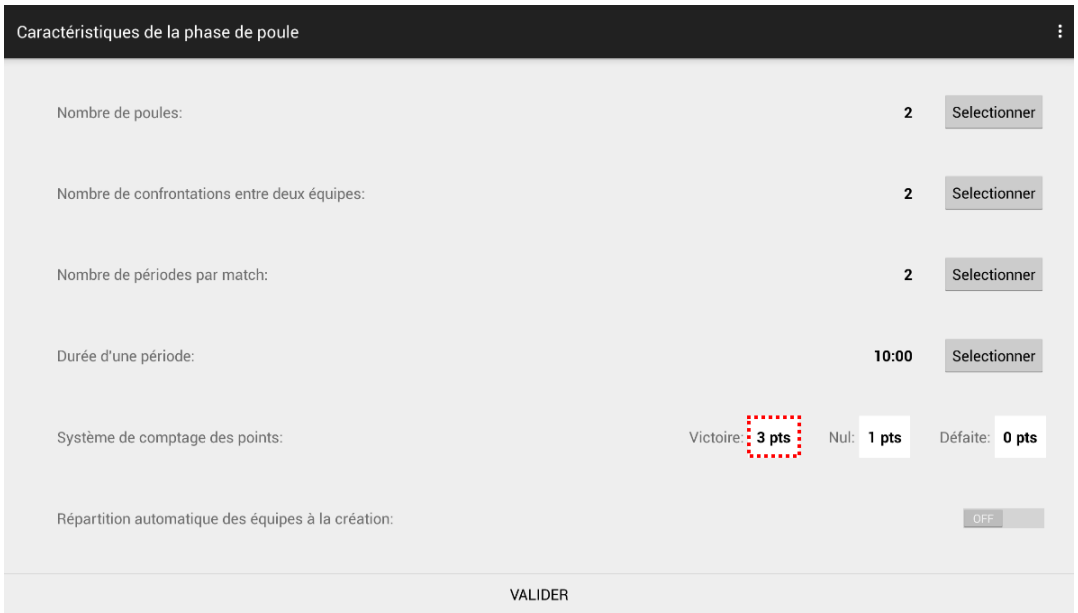
Il est possible de créer et de modifier un classement pour le tournoi dans la partie droite de l'écran de la vue d'ensemble.



5.3 Création d'une nouvelle phase

5.3.1 Phase de poule

Options



Nombre de poules

Nombre de confrontations: Le nombre de fois où deux équipes d'une poule seront confrontées l'une à l'autre.

Nombre de périodes et durée de chaque période

Systèmes de comptage de points: Le nombre de points à attribuer pour une victoire, un match nul ou une défaite. Pour changer les valeurs il suffit de cliquer sur l'encadré blanc qui contient la valeur définie.

Répartition automatique des poules: Si l'option est activée toutes les poules seront pré-remplies automatiquement. Il sera toujours possible de modifier l'appartenance d'une équipe à une poule.

Répartition des équipes

Il est possible d'ajouter une équipe à une poule en cliquant sur le bouton add. Les poules peuvent être sélectionnées en cliquant sur les flèches ⊂ et ⊃. Il est possible de supprimer une équipe d'une poule. Une fois que la sélection est validée, il n'est pas possible de modifier la répartition des équipes pour une poule.

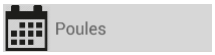
5.3.2 Phase tableau

Options

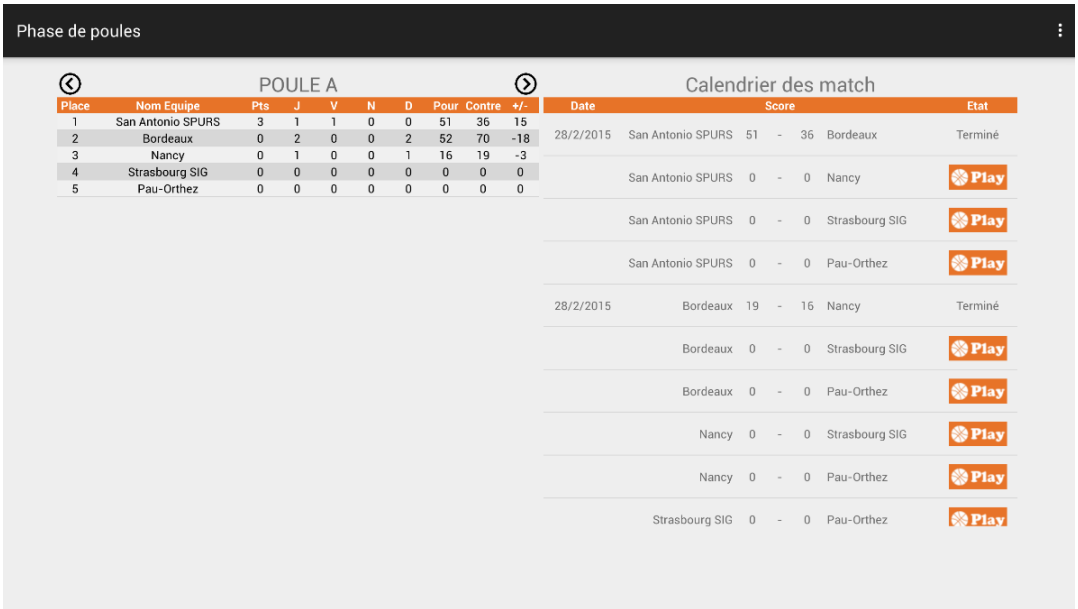
Nombre d'équipes: Le nombre d'équipes participants à la phase.

Petite finale: Indique si une petite finale sera jouée pour les perdants des demi-finale.

5.4 Phase de poule

Les phases de poule sont représentées par 

dans la liste des phases



Place	Nom Equipe	Pts	J	V	N	D	Pour	Contre	+/-
1	San Antonio SPURS	3	1	1	0	0	51	36	15
2	Bordeaux	0	2	0	0	2	52	70	-18
3	Nancy	0	1	0	0	1	16	19	-3
4	Strasbourg SIG	0	0	0	0	0	0	0	0
5	Pau-Orthez	0	0	0	0	0	0	0	0

Date	Score					Etat
28/2/2015	San Antonio SPURS	51	-	36	Bordeaux	Terminé
	San Antonio SPURS	0	-	0	Nancy	Play
	San Antonio SPURS	0	-	0	Strasbourg SIG	Play
	San Antonio SPURS	0	-	0	Pau-Orthez	Play
28/2/2015	Bordeaux	19	-	16	Nancy	Terminé
	Bordeaux	0	-	0	Strasbourg SIG	Play
	Bordeaux	0	-	0	Pau-Orthez	Play
	Nancy	0	-	0	Strasbourg SIG	Play
	Nancy	0	-	0	Pau-Orthez	Play
	Strasbourg SIG	0	-	0	Pau-Orthez	Play

Calendrier des matchs

Date: La date du match est fixée une fois le match terminé.

Score: Il est fixé à 0-0 pour un match non joué.

Play: Un clic sur ce bouton permet de lancer le match sélectionné. Pour importer le résultat du match depuis un fichier `.json` il faut rester appuyé sur le bouton.

Classement

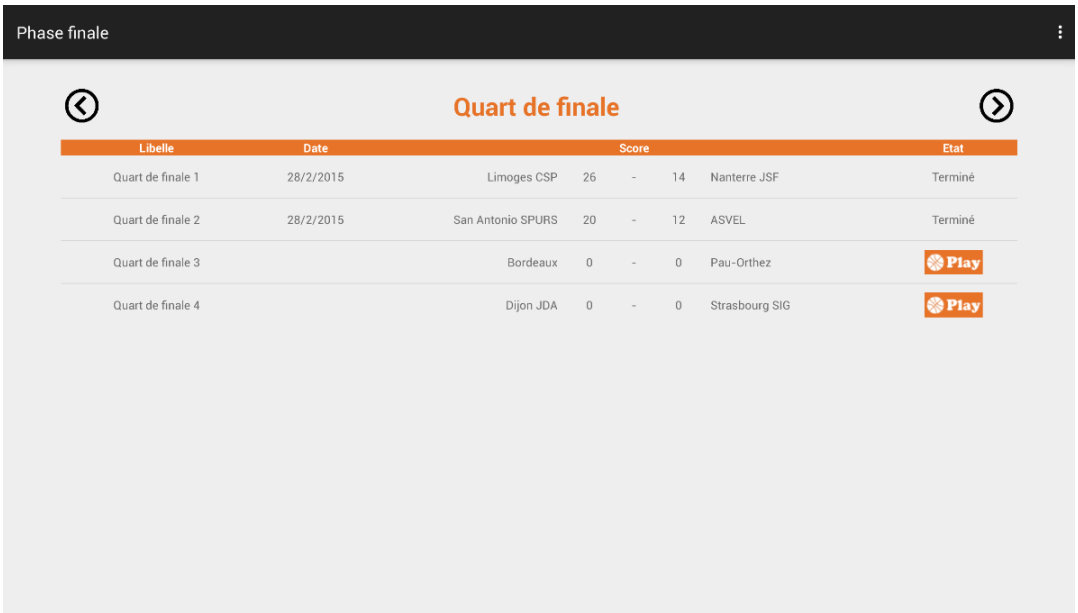
Le classement prends en compte les données suivantes.

Pts: Nombre de points engrangés.
J: Nombre de matchs joués.
V: Nombre de victoires.
N: Nombre de matchs nuls.
D: Nombre de défaites.
Pour: Nombre de paniers marqués.
Contre: Nombre de panier encaissés.
+/-: Différence entre le nombre de paniers marqués et le nombre de paniers encaissés.

Il est possible de naviguer d'une poule à l'autre en utilisant les flèches ⊲ et ⊳. Lors de la navigation le calendrier des matchs change automatiquement. Une fois que tous les matchs ont été joué pour la poule, un bouton **AJOUTER CLASSEMENT** apparaît et permet de générer automatiquement un classement à partir des résultats de la poule.

5.5 Phase finale

Les phases de tableau sont représentées par Finales dans la liste des phases



Il est possible de naviguer d'une phase finale à l'autre en utilisant les flèches ⊲ et ⊳. De même que pour les matchs de poule, un bouton **AJOUTER CLASSEMENT** apparaît.

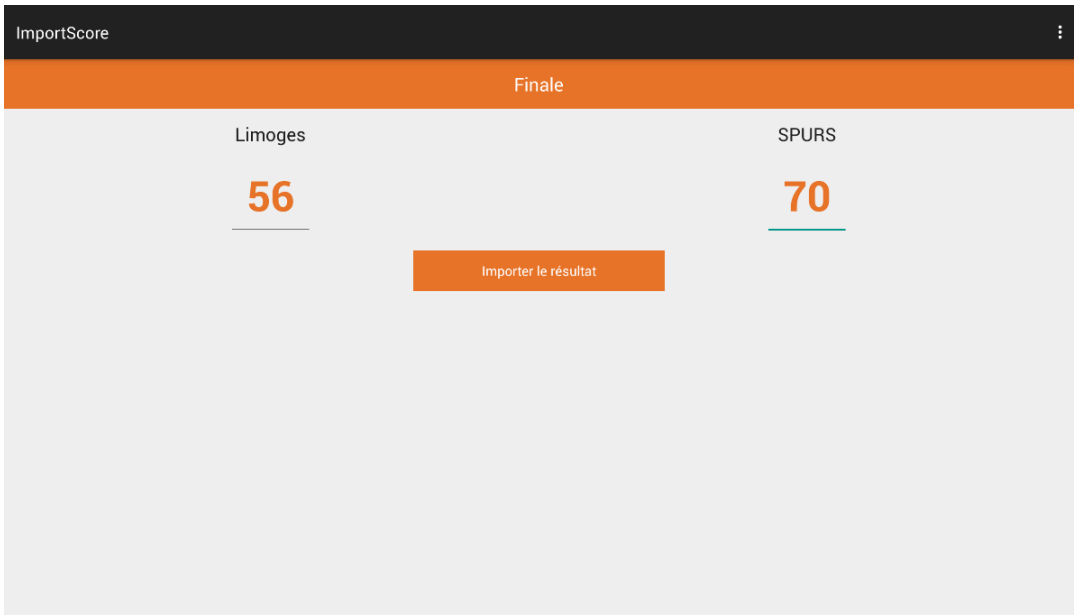
Sélection des équipes

Au début de la phase il suffit de cliquer sur le match afin de sélectionner les deux équipes qui vont s'affronter pour ce match. À noter que la liste des équipes est mise à jour dynamiquement en fonction des équipes disputant déjà un match.

Modification des équipes

Avant de lancer un match il est possible de modifier les équipes sélectionnées en cas d'erreur.

5.6 Import du résultat match



Il est possible d'importer le résultat d'un match directement depuis le mode tournoi. Pour cela il suffit de laisser le doigt sur le bouton

. Une boite de dialogue demandant de rentrer le résultat du match apparaît alors.

BASTATS

6. Import et export

Les fichiers d'import/export seront lus et écrits dans le dossier `Android/data/BastatsFiles` de la tablette.

6.1 Import d'un équipe

L'import d'une équipe se fait lors de la création du tournoi comme expliqué dans la partie 5.1. La structure du fichier est décrite ci-dessous. Un tableau d'équipes est composé d'objets représentant une équipe avec certains des attributs de la table **Equipe** (l'identifiant est celui du serveur) ainsi qu'un tableau de joueurs qui possède également des objets représentant un joueur avec les attributs **id**, **nom** , **prénom** et **pseudo**

```
 {
    "equipes": [
        { "id": "1",
          "couleur": "rouge",
  "photo":"a",
  "nom": "lakers",
          "Joueurs" :[
      { "id": "1",
"nom": "bryant",
"prenom": "kobe",
"pseudo": "black Mamba"
      }
      ,
      {  "id": "2",
 "nom": "Randle",
 "prenom": "Julius",
```

```
 "pseudo": "JR"
      }
  ]
}
,

{ "id": "2",
          "couleur": "bleu",
  "photo":"b",
  "nom": "spurs",
          "Joueurs": [

      { "id": "3",
"nom": "Duncan",
"prenom": "Tim",
"pseudo": "Timmy",
      }
      ,
      {  "id": "4",
 "nom": "Parker",
 "prenom": "Tony",
 "pseudo": "TP",
      }

  ]
}

    ]


}
```

6.2 Export complet d'un match

La structure du fichier `.json` d'export d'un match est décrite ci-dessous. L'export se lance depuis la vue d'ensemble d'un match et permet de récupérer toutes les statistiques saisies au cours du match.


```
   {
    "match":{
"_ID": "15" ,
"LIBELLE": "lakers vs spurs" ,
"DATE": "29/2/2015" ,
"FORMATION1_ID": "59" ,
"FORMATION2_ID": "60" ,
"REGLE_FORMATION1": "null" ,
"REGLE_FORMATION2": "null" ,
"RESULTAT": "1" ,
"SCORE1": "5" ,
"SCORE2": "0" ,
"PHASE_ID": "-2" ,
"ARBITRE1": "null" ,
"ARBITRE2": "null" ,
"ID_SERVEUR": "-1" ,
"equipeA":
{
    "actions": [
{"TypeAction" :{"Type:": "SHOOT" ,
"PTS": "3" ,
"REUSSI": "1"
       },
 "action" :{"JOUEUR_ACTEUR_ID": "1" ,
    "COMMENTAIRE": "" ,
    "TYPE": "0"
   },
 "tempsDeJeu" :{"MATCH_ID": "15" ,
"NUMERO_QT": "1" ,
"CHRONOMETRE": "00:07" ,
"NB_FAUTE_A": "0" ,
"NB_FAUTE_B": "1" ,
"LIBELLE": "null"
       }
},
{"TypeAction" :{"Type:": "SHOOT" ,
"PTS": "2" ,
"REUSSI": "1"
       },
 "action" :{"JOUEUR_ACTEUR_ID": "2" ,
    "COMMENTAIRE": "" ,
    "TYPE": "0"
```

```
   },
 "tempsDeJeu" :{"MATCH_ID": "15" ,
"NUMERO_QT": "1" ,
"CHRONOMETRE": "00:09" ,
"NB_FAUTE_A": "0" ,
"NB_FAUTE_B": "1" ,
"LIBELLE": "null"
       }
},
{"TypeAction" :{"Type:": "CONTRE" ,
"ACTION_ID": "65"
       },
 "action" :{"JOUEUR_ACTEUR_ID": "2" ,
    "COMMENTAIRE": "" ,
    "TYPE": "0"
   },
 "tempsDeJeu" :{"MATCH_ID": "15" ,
"NUMERO_QT": "1" ,
"CHRONOMETRE": "00:10" ,
"NB_FAUTE_A": "0" ,
"NB_FAUTE_B": "1" ,
"LIBELLE": "null"
       }
}]
    "formation":[{"formationJoueur" :{"JOUEUR_ID": "-1" ,
      "NUMERO": ""
     }
 }
 ,{"formationJoueur" :{"JOUEUR_ID": "-1" ,
       "NUMERO": ""
      }
  }
]
} ,
"equipeB":
{"actions":[
    {"TypeAction" :{"Type:": "FAUTE" ,
    "ACTION_ID": "62" ,
    "TYPE": "P"
   },
     "action" :{"JOUEUR_ACTEUR_ID": "4" ,
"COMMENTAIRE": "" ,
"TYPE": "0"
       },
     "tempsDeJeu" :{"MATCH_ID": "15" ,
    "NUMERO_QT": "1" ,
```

```
    "CHRONOMETRE": "00:03" ,
    "NB_FAUTE_A": "0" ,
    "NB_FAUTE_B": "0" ,
    "LIBELLE": "null"
   }
    },
    {"TypeAction" :{"Type:": "CONTRE" ,
    "ACTION_ID": "66"
   },
     "action" :{"JOUEUR_ACTEUR_ID": "4" ,
"COMMENTAIRE": "" ,
"TYPE": "0"
       },
     "tempsDeJeu" :{"MATCH_ID": "15" ,
    "NUMERO_QT": "1" ,
    "CHRONOMETRE": "00:13" ,
    "NB_FAUTE_A": "0" ,
    "NB_FAUTE_B": "1" ,
    "LIBELLE": "null"
   }
    }]
 "formation":[{"formationJoueur" :{"JOUEUR_ID": "-1" ,
   "NUMERO": ""
  }
      }
      ,{"formationJoueur" :{"JOUEUR_ID": "-1" ,
    "NUMERO": ""
   }
       }
     ]
}
    }
}
```


Un objet match contient en plus des attributs de **Match** les objets **équipe A** et **équipe B** comportant un tableau d'actions incluant une action, son temps de jeu et son type associé, ainsi qu'un tableau de formations où chaque joueur est représenté par son identifiant sur le serveur, et son numéro de maillot pour le match.

6.3 Génération d'un PDF

Depuis la vue d'ensemble d'un match il est possible de générer un fichier `.pdf` dans le répertoire`Android/data/BastatsFiles`. Le nom du fichier suit la structure suivante : `date_equipeAVSequipeB.pdf`.
Pts: Nombre de points engrangés.
Fg: Nombre de tirs (Field goals).

Ft: Nombre de de lancer francs (Free Throw).
3P: Nombre de 3 points.
Off: Nombre de rebonds offensive.
Def: Nombre de rebonds défensive.
Reb: Nombre total de rebonds.
+/-: Différence entre le nombre de paniers marqués et le nombre de paniers encaissés.
TOT: Nombre total de tirs.
A: Nombre de passes décisives (assists).
Pf: Nombre de fautes personnel (personnal fouls).
To: Nombre de pertes de balles.

lakers 14 - 12 spurs

	lakers															
No	Player	MIN	FG	3P	Ft	+/-	Off	Def	Reb	A	To	Stl	Bs	Ba	PF	Pts
1	bryant	00:00	2-1	0-0	1-0	6	0	1	1	0	0	0	1	1	2	5
2	randle	00:00	2-1	1-0	0-0	6	0	0	0	0	0	0	0	0	0	7
3	howard	00:00	0-1	0-0	0-0	1	0	1	1	0	0	0	1	1	1	0
Eq	JoueurNeutre	00:00	1-0	0-0	0-0	4	0	1	1	0	0	0	1	1	0	2

	spurs															
No	Player	MIN	FG	3P	Ft	+/-	Off	Def	Reb	A	To	Stl	Bs	Ba	PF	Pts
4	parker	00:00	2-0	0-0	0-0	6	1	1	2	0	0	0	0	0	2	4
5	diaw	00:00	2-1	0-0	0-0	9	3	1	4	0	0	0	2	2	4	4
6	duncan	00:00	1-1	0-0	0-0	3	0	1	1	0	0	0	1	1	1	2
Eq	JoueurNeutre	00:00	1-0	0-0	0-0	5	1	1	2	0	0	0	1	1	0	2

Arbitre 1 : null Abitre 2 : null

C. Dessins papiers

Menu Bar | Amical

Spinner
- Sélectionner Equipe
- Ajouter Equipe

V.S.

Couleur Maillot
(afficher valeur par défaut, rien sinon)

Time +
Nb QT

CONFIRMER

Sélection Joueur

Equipe A	Equipe B
Joueur 1	Joueur 1
Joueur 2 ✓	Joueur 2
Joueur 3	Joueur 3
Joueur 4 ✓	⋮
⋮ ✓	⋮
⋮	⋮

Créer Joueur | Next | Créer Joueur (de équipe)

Sélection Numéro Maillot

Equipe A	Equipe B
2 Joueur 1	
8 Joueur 3	
3 Joueur 4	

OnClick SelectedItem

clicker pour sélectionner numéro

START

Boite de Dialogue:

Numéro Maillot

Num Maillot (Edit Text)

Valider

On supprime les numéros déjà sélectionné

Sélection Tournoi

Vu poule X

Calendrier des Matchs

Journée 1

A	26 - 32	B	✓
C	30 - 32	D	✓

Journée 2

A	-	C	Play
B	-	D	Play

Poule A

Equipe	J	Pts	W	D	L	...
A						
B						
C						
D						

Menu latéral

- Nom du tournoi
- Poule A
- Poule B
- Poule C
- ⋮
- Phases finales
- Statistiques

Spécifique tournoi phases de poules

Click sur un match = édition score etc...
Click sur play: lance écran sélection équipe en verrouillant les 2 équipes mais choix couleur maillot.
Puis possibilité sélectionner joueurs qui vont jouer le match

Tournois

- Nouveau Tournoi
- Continue
- Voir résultats
- Import/Export

= Juste génération ds BDD à la fin c'est équivalent à cliquer sur reprendre avec le tournoi tt juste créer

Caractéristiques du tournois

Nom tournoi _____

Date de début _____

Elimination directe ou Poules + phase finale

Elimination Directe

Nombre période : [ ]

Durée période : [ ]

Match Aller/Retour ○

Match unique ◉

Tableaux automatique ◉

Tableau manuel ○

Valider

Elimination directe

Sélectionner Equipe

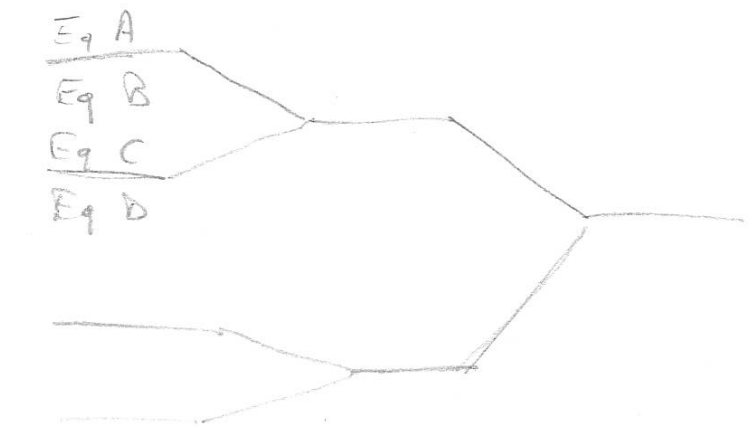
Ajouter Equipe

Nombre d'équipes :

VALIDER

× × × × × × × × × × ×

Elimination Directe

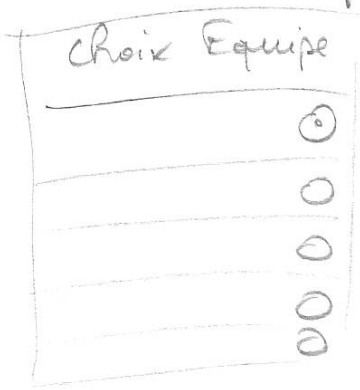


retour

VALIDER

MODIFIER

click sur nom équipe + proposition pour swap avec autre équipe



Lance l'écran REPRENDRE avec l'id du tournoi créé.

Génère le calendrier des matchs dans le même temps

Tournoi phase de poule

Poules

Nb QT :
Temps QT :

Aller/Retour
Aller

Génération automatique
Gestion manuelle

Continuer

Phases finales

idem

Sélection des équipes

Nb equipes :
(Algo génération auto des poules fct² nb équipe)

Manuelle | Auto

Liste équipes

Equipe A Add
Add
Add
Add
Add

Poule X

Click équipe ds liste équipe
↳ Ajout ds poule X

Click ds Poule X sur ☒
↳ Remis équipe dans liste équipe

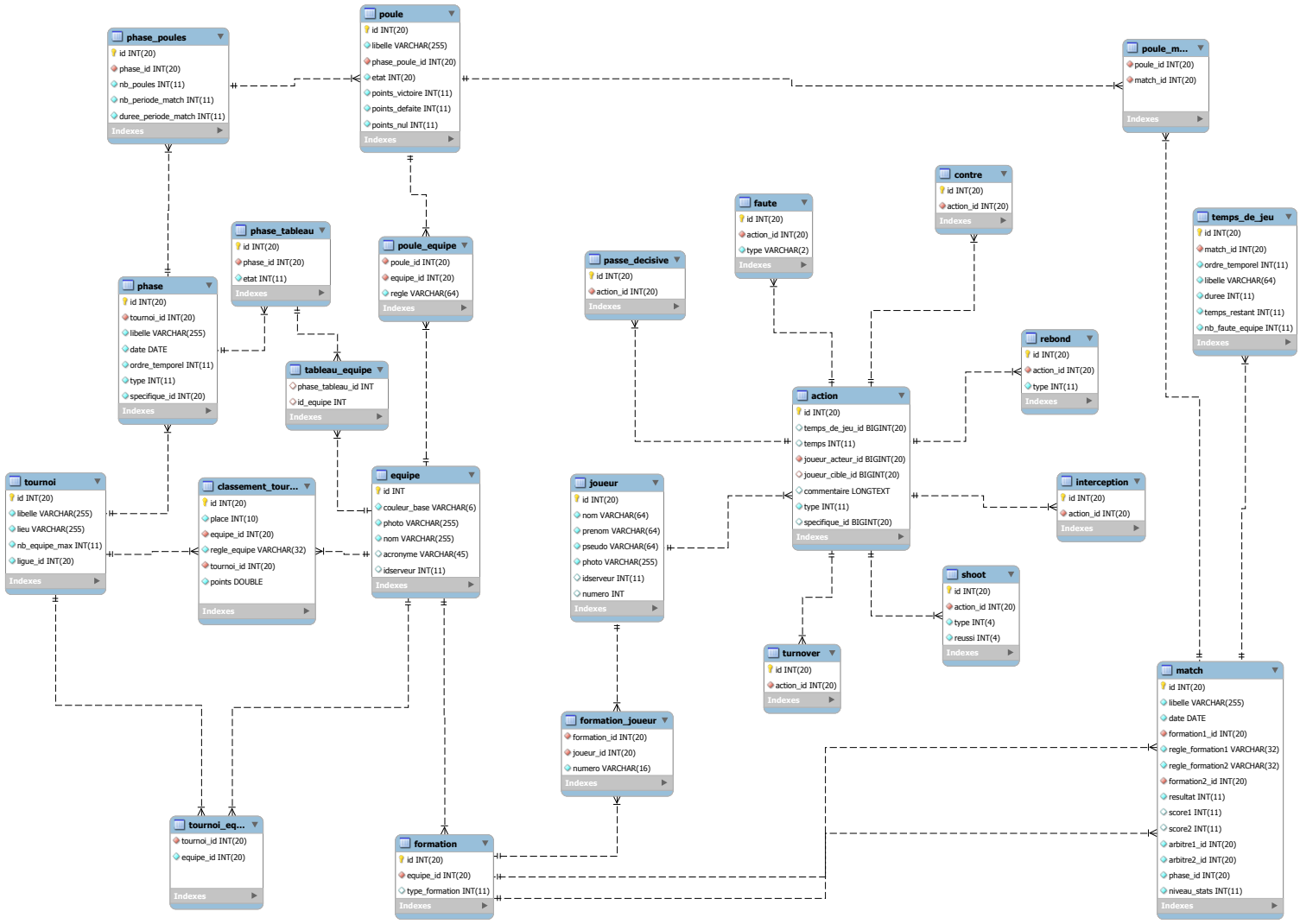
Valider → envoi sur écran reprendre et générer calendrier
(ou sur sélection des confrontations prévues pour les phases finales)

Valider

Pour Auto : liste eq vide, poule pleines, possibilité modif

BASTATS

D. Schéma de la base de données


phase_poules
id INT(20)
phase_id INT(20)
nb_poules INT(11)
nb_periode_match INT(11)
duree_periode_match INT(11)
Indexes
poule
id INT(20)
libelle VARCHAR(255)
phase_poule_id INT(20)
etat INT(20)
points_victoire INT(11)
points_defaite INT(11)
points_nul INT(11)
Indexes
poule_m...
poule_id INT(20)
match_id INT(20)
Indexes
contre
id INT(20)
action_id INT(20)
Indexes
faute
id INT(20)
action_id INT(20)
type VARCHAR(2)
Indexes
temps_de_jeu
id INT(20)
match_id INT(20)
ordre_temporel INT(11)
libelle VARCHAR(64)
duree INT(11)
temps_restant INT(11)
nb_faute_equipe INT(11)
Indexes
phase_tableau
id INT(20)
phase_id INT(20)
etat INT(11)
Indexes
poule_equipe
poule_id INT(20)
equipe_id INT(20)
regle VARCHAR(64)
Indexes
passe_decisive
id INT(20)
action_id INT(20)
Indexes
phase
id INT(20)
tournoi_id INT(20)
libelle VARCHAR(255)
date DATE
ordre_temporel INT(11)
type INT(11)
specifique_id INT(20)
Indexes
tableau_equipe
phase_tableau_id INT
id_equipe INT
Indexes
rebond
id INT(20)
action_id INT(20)
type INT(11)
Indexes
action
id INT(20)
temps_de_jeu_id BIGINT(20)
temps INT(11)
joueur_acteur_id BIGINT(20)
joueur_cible_id BIGINT(20)
commentaire LONGTEXT
type INT(11)
specifique_id BIGINT(20)
Indexes
tournoi
id INT(20)
libelle VARCHAR(255)
lieu VARCHAR(255)
nb_equipe_max INT(11)
ligue_id INT(20)
Indexes
classement_tour...
id INT(20)
place INT(10)
equipe_id INT(20)
regle_equipe VARCHAR(32)
tournoi_id INT(20)
points DOUBLE
Indexes
equipe
id INT
couleur_base VARCHAR(6)
photo VARCHAR(255)
nom VARCHAR(255)
acronyme VARCHAR(45)
idserveur INT(11)
Indexes
joueur
id INT(20)
nom VARCHAR(64)
prenom VARCHAR(64)
pseudo VARCHAR(64)
photo VARCHAR(255)
idserveur INT(11)
numero INT
Indexes
interception
id INT(20)
action_id INT(20)
Indexes
shoot
id INT(20)
action_id INT(20)
type INT(4)
reussi INT(4)
Indexes
turnover
id INT(20)
action_id INT(20)
Indexes
match
id INT(20)
libelle VARCHAR(255)
date DATE
formation1_id INT(20)
regle_formation1 VARCHAR(32)
regle_formation2 VARCHAR(32)
formation2_id INT(20)
resultat INT(11)
score1 INT(11)
score2 INT(11)
arbitre1_id INT(20)
arbitre2_id INT(20)
phase_id INT(20)
niveau_stats INT(11)
Indexes
formation_joueur
formation_id INT(20)
joueur_id INT(20)
numero VARCHAR(16)
Indexes
tournoi_eq...
tournoi_id INT(20)
equipe_id INT(20)
Indexes
formation
id INT(20)
equipe_id INT(20)
type_formation INT(11)
Indexes