****

**LO52**

**RAPPORT DE PROJET**

**-**

**APPLI F1 RACE**

***Auteurs :***  ***Encadrants :***

GAUJOUR Bastien BRISSET Fabien PALERMO Enzo ROMET Pierre

**SOMMAIRE :**

I – Cahier des charges :

* *Pages 3 à 4 : Détails du cahier des charges*

II – Application :

* *Pages 4 à 8 : Interface de l'application*
* *Pages 8 à 9 : Détails de l'algorithme de création des équipes*
* *Pages 9 à 10 : Détails de l'utilisation des bases de données*
* *Pages 10 à 11 : Fonctionnalités manquantes et améliorations possibles*

III – Conclusion :

* *Page 11*

**Cahier des charges :**

Classement/Statistiques :

* Meilleur temps par coureur :
  + Au Pitstop 1
  + Au Fractionné
  + Au Sprint
  + Au Cycle
* Meilleur temps par équipe :
  + Au Pitstop 1
  + Au Fractionné
  + Au Sprint
  + Au Cycle
  + En temps total (Somme des cycles ou des ateliers)

Bonus : Fonction historique des courses précédentes

Chronométrage :

* Manuel :
  + 1 Appui pour le départ d’un coureur
  + 1 appui pour le split de chacun de ses ateliers.

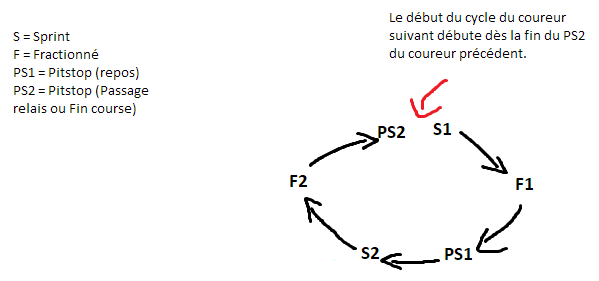
Il y a donc un chrono manuel par équipe.

* Auto (Bonus):
  + Chrono général qui démarre au départ du coureur 1 et qui s’arrête au dernier split du dernier coureur. Indique le temps total de la course

**Attention :** Les équipes peuvent concourir en même temps, il peut donc y avoir x chrono manuels simultanés si x équipes.   
(Pas de gestion du multitouch cependant donc les départs seront forcément différés par l'utilisateur.)

Setup de la course :

* Rentrer manuellement chaque joueur, avec nom, prénom, niveau (de 1 à 100).  
  Le nombre total de participant devra être un multiple de 3 (indiquer le nombre total en direct à l’user)
* Générer des équipes équilibrées de 3 coureurs, de telle façon que l’écart entre l’équipe la plus forte et la plus faible ne dépasse pas 25.
* Rentrer manuellement l’ordre de passage des équipes, et des coureurs dans chaque équipe



**Application :**

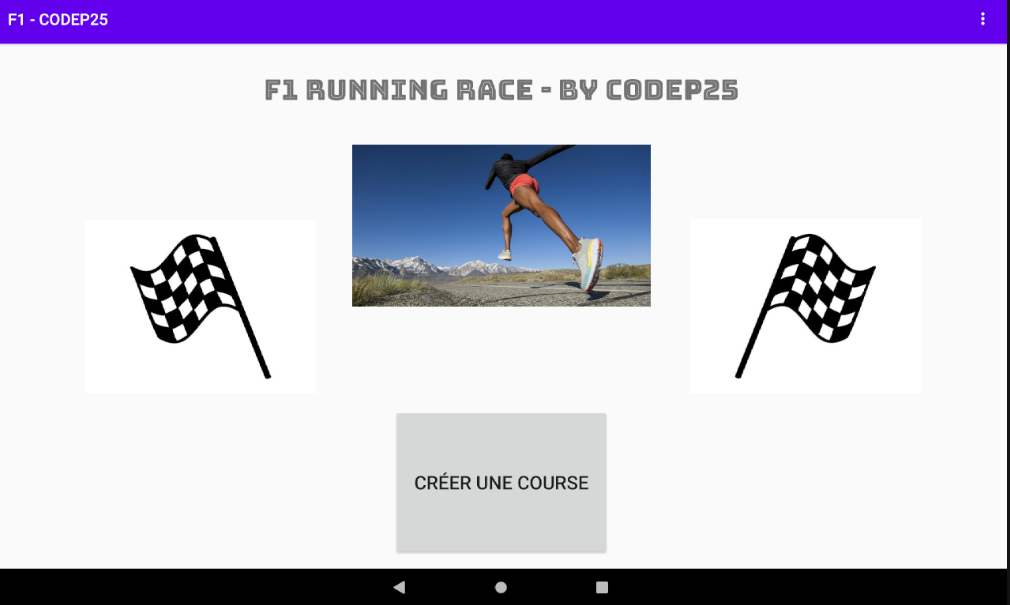
Nous avons fait le choix d'utiliser kotlin pour cette application, non seulement parce qu'il est plus pratique que Java, mais aussi parce qu'il est son successeur qui finira par le remplacer à l'avenir.  
  
Il nous a donc semblé interessant d'apprendre à utiliser kotlin grâce à ce projet.

Interface de l’application :

L'application se décompose en plusieurs fragments correspondants chacun à un écran.   
Dans l'ordre, cela donne :

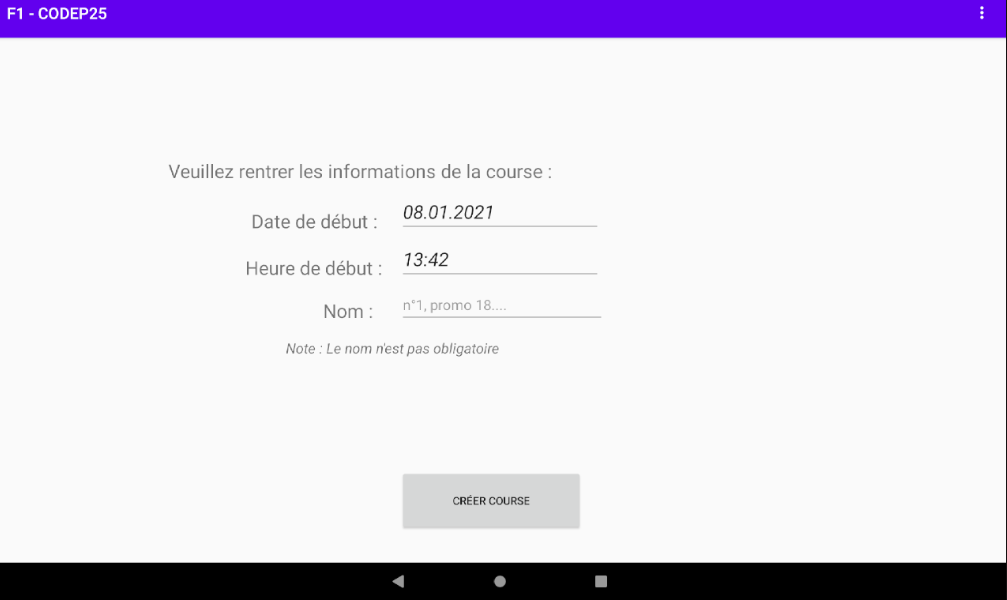
**1 - Présentation :**

* Bouton création course : Passe à l'écran suivant

****

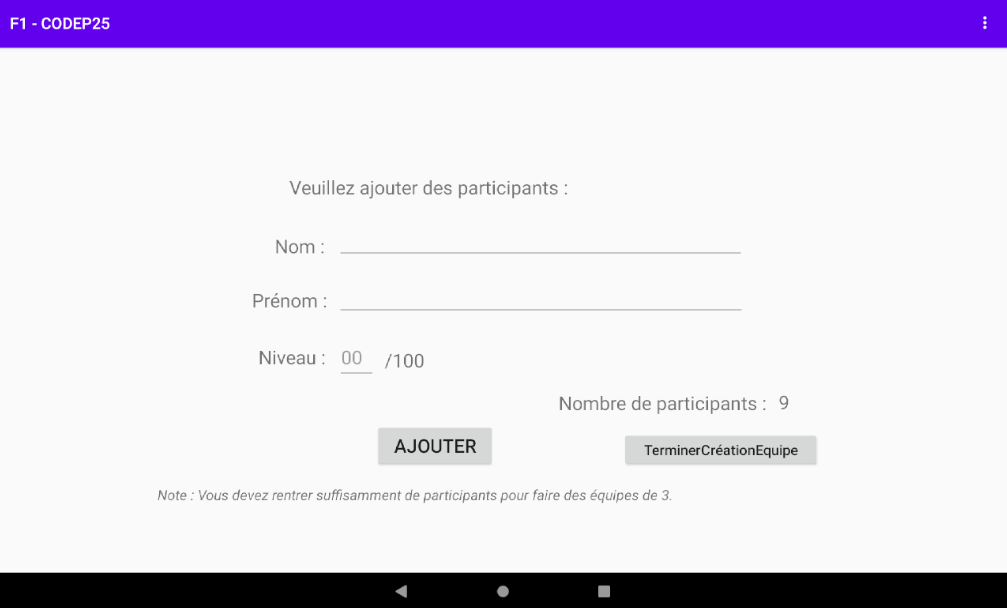
**2 - Création course :**

* Date : les infos sont pré-remplies au moment de laffichage
* Heure : les infos sont pré-remplies au moment de laffichage
* Nom : Le nom n'est pas obligatoire, on pourra identifier la course par sa date decréation par la suite.
* Bouton "*Créer une course*" : Créer la base de données course et passe à l'écran suivant.



**3 - Création des participants :**

* Nom/Prénom/Niveau
* Bouton "*Ajouter*" : Ajoute le participant à la base de données
* Bouton "*TerminerCréationEquipe*" : Vérifie qu'il y ait le bon nombre de participants et lance la création d'équipe.

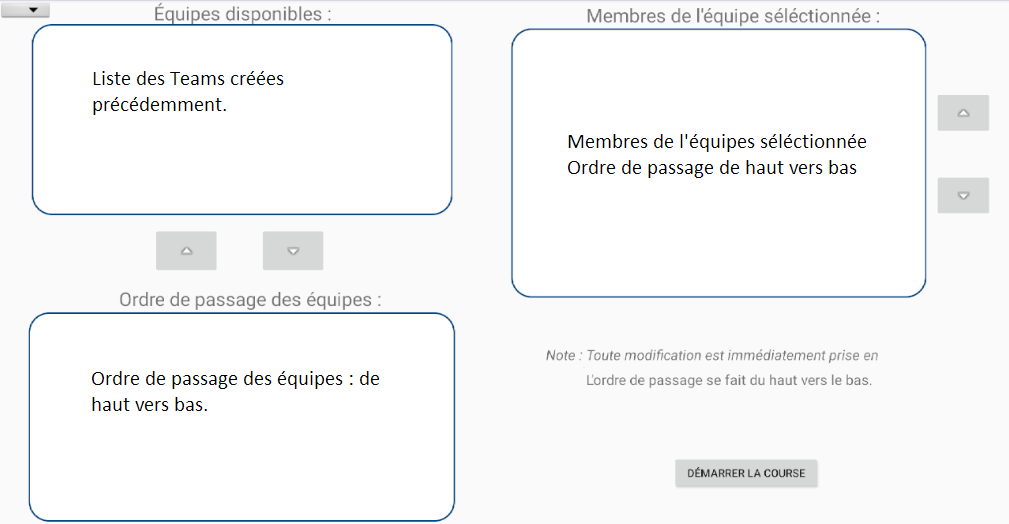


**3.bis – Génération des équipes :**

La génération des équipes se fait en background entre la transition des écrans 3 et 4. Si l'algorithme met trop de temps, on pourra rajouter une indication de chargement à l'user pour qu'il n'ai pas de doute sur le fonctionnement de l'application.

**4 - Gestion des ordres de passage :**

* Ordre de passage des coureurs dans une équipe : On séléctionne une équipe dans le cadre des équipes disponibles, ce qui affiche les joueurs qui la composent dans le cadre de droite. On peut alors modifier leur ordre de passage avec les flèches à coté ducadre.
* Ordre de passage des équipes : Une fois que l'on a vérifié l'ordre de passage des joueurs, on fait passer l'équipe séléctionnée dans le cadre du bas ce qui valide son ordre de passage (reste annulable avec la flèche vers le haut).

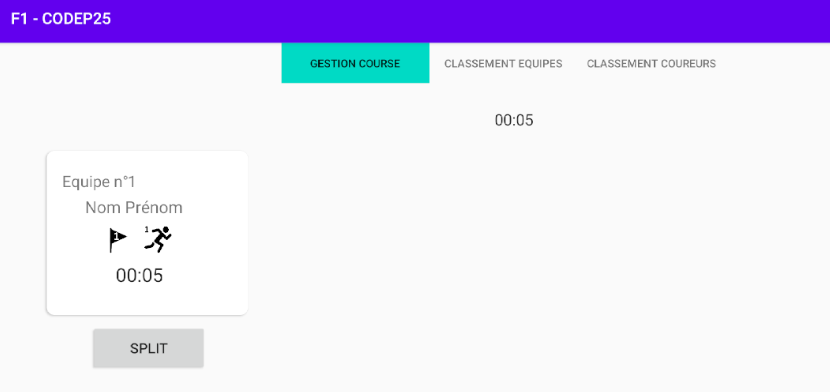


**Gestion de la course:**

* Onglet Gestion course :
  + Chrono général (en haut) : du premier départ à la dernière arrivée
  + Team boxes : Num d'équipe, nom coureur, atelier en cours, chrono de l'équipe.  
    *(A chaque clic sur le bouton split, on récupère le temps de l'atelier. Le premier clic sonne le départ de l'équipe.)*
* Onglet Classement équipes :
  + Accès au classement par équipe (Par temps total)
* Onglet Classement coureurs :
  + Accès au classement par coureur (Par temps de cycle)

Dans les onglets de classement, nous pourrons avoir accès aux infos d'une équipe/coureurs lors du clic :

* Clic équipe (depuis classement équipe) : coureurs qui la composent + leurs temps par cycle.
* Clic coureur (depuis classement coureur) : leurs temps par atelier + leur numéro d'équipe.



Algorithme de création des équipes :  
  
Il est (à priori ?) impossible de trouver une formule qui satisfait la condition à coup sur quel que soit le tirage, puisque la construction de la dite formule se fait par rapport à un tirage donné. Nous avons donc essayé de modifier les valeurs après une première tentative de création la plus équilibrée possible.  
  
L'algorithme peut se décomposer en plusieurs étapes :

*Etape 1 : Création des premières équipes :*  
  
Sois une liste de valeurs des niveaux des joueurs, ordonnée par ordre croissant :  
16,20,33,35,36,56,57,65,66,69,75,80,81,90,95,98,99,100  
  
On forme les équipes de la façon suivante :  
 - On prend le *plus petit* élément, le *plus grand* élément et l'élément *médian* (En cas de nombre pair, élément *médian -1)*  
 - On forme l'équipe n°1 avec ces 3 niveaux et on les retire de la liste (Ex ici équipe n°1 = 16 + 100 + 66 = 182)  
 - …etc jusqu'à vider la liste

*Etape 2 : Premier test de l'écart :*Ensuite, on créé une liste avec les valeurs totales de chaque équipe, puis on l'ordonne par ordre croissant, et on redéfini le numéro de l'équipe selon la position dans cette liste.  
Dans notre exemple cela donne 182, 183, 188, 196, 205, 217  
  
On réalise un test de l'écart entre la *plus forte* et la *plus faible*, ici il est de 35 et dépasse donc la valeur seuil de 25.  
On passe alors au premier ajustement.

*Etape 3 : Ajustement n°1 :*  
  
On prend donc l'équipe *médiane +1* (ici 196, équipe n°4) et l'équipe la plus forte (ici 217, équipe n°6), puis on échange leurs valeurs médiane :  
Avant échange :   
 E4 = 33 + 98 + 65 = 196  
 E6 = 56 + 81 + 80 = 217  
Après échange :  
 E4 = 33 + 98 + **80 = 211**  
 E6 = 56 + 81 + **65 = 202**

On ré-ordonne les équipes et on redéfini leurs positions :  
182, 183, 188, 202, 205, 211  
Puis on reteste l'écart : il est de 29.  
  
*Etape 3 : Ajustement n°2 :*

On continue donc l'échange de valeur médiane, mais cette fois-ci entre la valeur *médiane -1* (188, E3) et la *plus forte* (211, E6).  
(Il a fallu utiliser médiane -1 et non médiane +1 car les valeurs 202 et 211 correspondaient déjà aux deux équipes récemment créées, il y aurait donc eu une étape inutile qui aurait annulé l'échange précédent)  
Avant échange :  
 E3 = 20 + 99 + 69 = 188  
 E6 = 33 + 98 + 80 = 211  
Après échange :  
 E3 = 20 + 99 + **80** **= 199**  
 E6 = 33 + 98 + **69 = 200**Nous avons donc le nouveau classement 182, 183, 199, 200, 202, 205  
Ce qui nous donne un écart de 23 < 25.

On réalise autant d'ajustement que nécessaire jusqu'à réduction de l'écart sous le niveau des 25.

Bases de données :  
  
Afin de stocker de façon permanente les données de l'application, il est nécessaire d'utiliser des bases de données.  
  
Cette fonctionnalité nous sera notamment utile pour consulter les différents classements, mais aussi pour consulter l'historique des précédentes courses.  
Cela permet également de clarifier la gestion des données de l'application dans le programme.

Nous utiliserons la fonctionnalité **Room** d'android studio qui permet la gestion de bases de données facilement en intéragissant avec des objets.

Il y aura donc 3 bases de données différentes :

*Race db :*

Cette db contiendra les champs suivants : *ID | RaceName | DateCreation | HeureCreation | Teams db | Runners db*   
  
Les champs "Teams db" et "Runners db" serviront à stocker les 2 databases suivantes afin de consulter l'historique des courses.

La db sera initiée sur l'écran n°2.

*Runners db :*

Cette db contiendra les champs suivants : *ID | Nom | Prenom | Niveau | Order | tps 1 | tps 2 | tps 3 | tps 4 | tps 5 | tps cycle*  
  
Les champs "tps 1…5" et "tps cycle" serviront aux fonctionnalités de classements et d'historique.  
Le champ "order" défini l'ordre de passage du joueur dans son équipe, de 1 à 3.

La db sera initiée sur l'écran n°3.

*Teams db :*

Cette db contiendra les champs suivants : *ID | Order | ID Runner 1 | LevelRunner1 | ID Runner 2 | LevelRunner2 | ID Runner 3 | LevelRunner3 | Niveau total | tps total*  
  
Le champ "tps total" sert lui aussi aux fonctionnalités de classement et d'historique.   
Le champ "order" défini l'ordre de passage de l'équipe, de 1 au nombre d'équipes.

La db sera initié après l'algorithme de création d'équipe, entre l'écran n°3 et 4.  
  
*Note :* Il a fallu ajouter le plugin "kapt" pour que les databases fonctionnenent avec kotlin. Il y avait un crash de l'application à la création.

Fonctionnalités manquantes et améliorations possibles :

Ce projet nous a présenté un certains nombre d'obstacle techniques dont certains que nous n'avons pas su franchir dans le temps imparti.  
  
Il manque donc les fonctionnalités suivantes :

* Lier la base de données *Race db* à ses bases de données *Teams db* et *Runners db* correspondantes
* Implémenter l'algorithme de création d'équipes
* Afficher des listes dynamiques pour la gestion de l'ordre de passage sur l'écran n°4 (RecyclerView, Adapter, ViewHolder…..?)
* Afficher les Teams Boxes selon les équipes sur l'écran n°5.
* Récupérer le temps des coureurs au split
* Afficher les classements en temps réel avec **Kotlin Co-Routines** (Marche mieux que LiveData pour les Room databases

**Conclusion :**

Ce projet à été l'occasion de découvrir plus en détail l'interface d'Android studio, d'intéragir avec les VM Android (Voir avec son propre smartphone), et d'apprendre à utiliser le langage Kotlin.

Le projet fût donc techniquement riche d'enseignements. Et bien que l'application soit incomplète, ce fut un grand plaisir de découvrir un peu plus le fonctionnement des applications que l'on cotoie au quotidien. Je (Bastien Gaujour) n'hésiterai donc plus à l'avenir à essayer de créer des applications pour mes besoins personnels.