F1 Levier

DIAGNE Mouhamed - SALL Ndeye Marie

Plan

- Besoin du client
- Planning et Organisation
- Algorithme utilisé
- Présentation de l’application
- Bilan

Besoin du client

- ➔ Créer des équipes équilibrées en fonction du nombre de participants et du niveau de chacun
- ➔ Déterminer le temps de chaque coureur et le temps total de chaque équipe
- ➔ avoir des statistiques individuelles (participants) ou équipes et déterminer l’équipe avec le cycle le plus rapide

Planning et Organisation

Planning

- Création des participants
- Création des équipes
- Classement
- Utilisation de Room

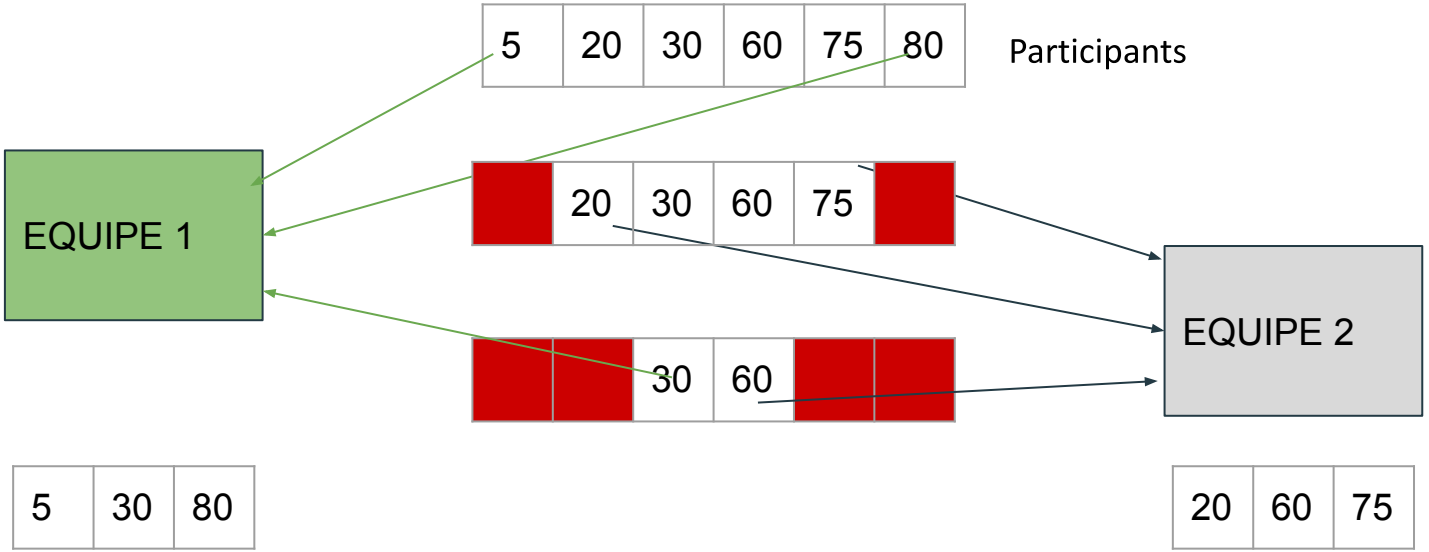
Organisation

- On se partageait les tâches
- Faire des recherches sur Room
- Réfléchir sur l'algorithme de création des équipes

Algorithme de génération des équipes

- ➔ Trier la liste des participants
- ➔ Itérer sur le nombre d’équipes
- ➔ Pour chaque itération
  - ◆ créer une équipe
  - ◆ ajouter le participant qui a le plus grand niveau et celui qui a le plus petit niveau à l’équipe
  - ◆ supprimer les participants affectés de la liste
- ➔ Ajouter à chaque équipe un dernier joueur

Exemple : Algorithme utilisé



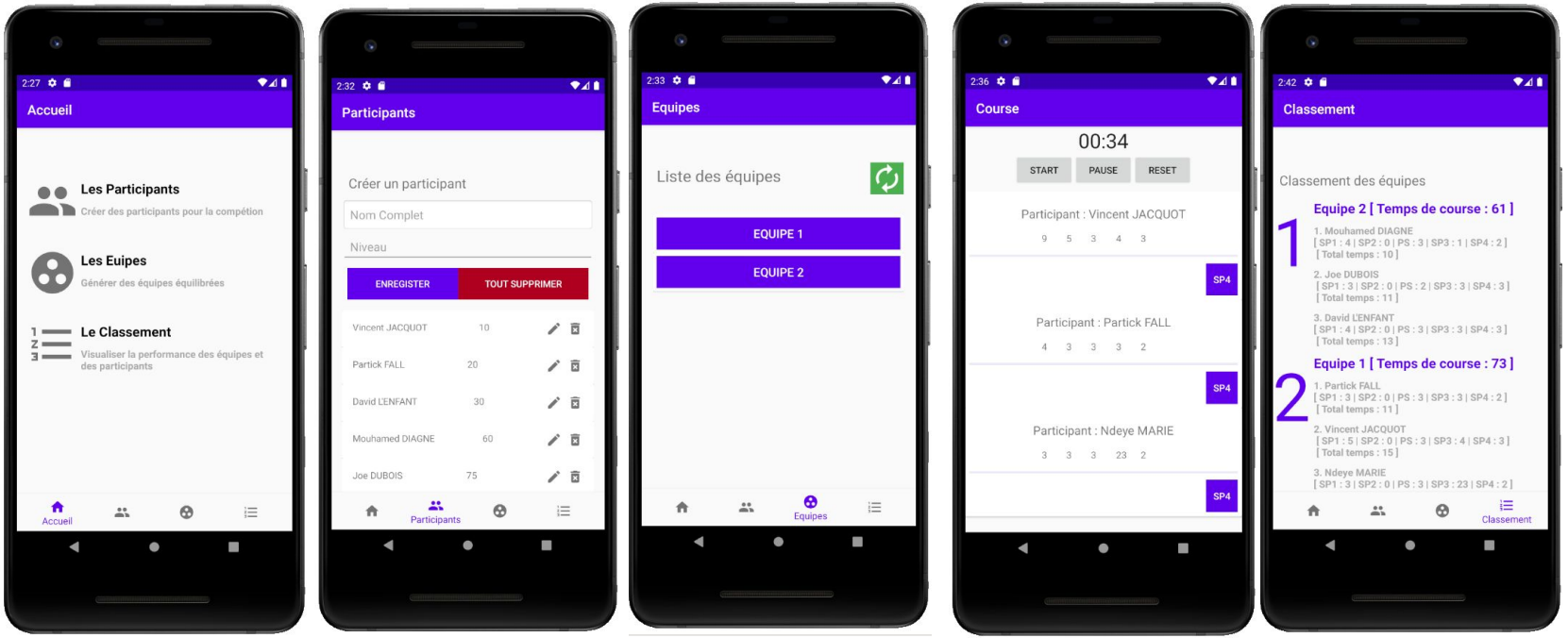
Ici permutation entre 30 et de 60, car la différence des niveaux > 25

Implémentation

```
// Algorithm to generate "equipe"
void participantDispatcher(List<Participant> participants) {
    int sizeEquipe = participants.size()/3, size, i = 0;

    for ( int z = 0 ; z < sizeEquipe; z++ ) {
        size = participants.size()-1;
        //Create "equipe"
        Equipe e = createEquipe(i);
        //Add last participant and first participant to the "equipe";
        database.participantDao().updateEquipe(participants.get(i).getID(),e.getID());
        database.participantDao().updateEquipe(participants.get(size).getID(),e.getID());
        //Remove affected "participants" from list of "participants"
        participants.remove(size);
        participants.remove(i);
        i++;

    }
    dataList = database.equipeDao().getAll();
    size = participants.size();
    for (int y=0; y<size;y++) {
        //Add one participant to the "equipe"
        database.participantDao().updateEquipe(participants.get(y).getID(), dataList.get(y).getID());
    }
    // Do "equipe" balancing
    equipeBlancing();
}
```

```
//Create one "equipe"
Equipe createEquipe(int i) {
    Equipe equipe = new Equipe();
    equipe.setName("Equipe " + (i+1));
    database.equipeDao().insert(equipe);
    return database.equipeDao().getByNom(equipe.getName());
}

//Function to balance "equipe" from "equipe" level
void equipeBlancing() {
    dataList = database.equipeDao().getAllByLevel();
    // Load lowest and highest "equipe"
    Equipe e1 = dataList.get(0);
    Equipe e2 = dataList.get(dataList.size()-1);
    // Load participants by each "equipe"
    List<Participant> pListE1 = database.participantDao().getAllByEquipeID(e1.getID());
    List<Participant> pListE2 = database.participantDao().getAllByEquipeID(e2.getID());
    // Verify if the difference of equipe Level  is higher than 25
    if (((e1.getLevel() !=null) && (e2.getLevel() !=null))) {
        if ((e2.getLevel() - e1.getLevel()) > 25 ) {
            // Switch participant index 2 and update database
            database.participantDao().updateEquipe(pListE1.get(1).getID(), e2.getID());
            database.participantDao().updateEquipe(pListE2.get(1).getID(), e1.getID());
        }
    }
}
```

Présentation de l'application



Bilan

- ➔ Toutes les fonctionnalités sont implémentés
- ➔ Améliorer l’interface de Course avec simulation d’une piste