



Rendu Final: Projet Informatique Individuel

db (danse bien)

UE Sciences fondamentales

SERNA HERNANDEZ Alexandra

2021-2022

Table de matières

Contexte et objectifs	2
Procédure d'installation	2
Résultat final	3
Choix techniques et architecture de l'application	8
Bilan des exigences	8
Pistes d'amélioration	8
Gestion de projet	8
Conclusion & Bilan personnel	9
Annexes	10
	13

I. Contexte et objectifs

Mon projet **db** (dansebien) servira à faciliter la gestion de la musique dans un cours de danse classique, tout en proposant une interface dans laquelle les professeurs puissent noter les exercices associés. S'il existe des outils numériques qui proposent des musiques pour des cours de danse classique, ils ne permettent pas aux professeurs d'enregistrer leurs propres combinaisons d'exercices. Je vais ainsi concevoir et développer une application mobile qui relie les deux.

La plateforme s'adresse principalement aux professeurs de danse classique mais elle a pour objectif de s'inscrire dans d'autres domaines de la danse, voire d'autres sports. De plus, même si elle ne se centrera que sur le cadre d'un cours de danse, nous pourrions imaginer des fonctionnalités liées à la préparation d'une présentation, grâce auxquelles l'on puisse enregistrer des chorégraphies, et où les professeurs puissent travailler de façon collaborative.

Ce rapport s'accompagne du code source de mon application.

II. Procédure d'installation

1. Téléchargez le dossier à cette adresse : [alexandrash31/dansebienP2I_VF](https://alexandrash31.github.io/dansebienP2I_VF)
2. Ouvrez le projet dans un éditeur de code (Visual Studio Code (VSC)), pour cela :
 - ouvrir VSC
 - cliquer sur Fichier puis Ouvrir
 - sélectionner le dossier **dansebienP2I_VF**
3. Ouvrez le terminal de VSC, placez-vous sur le **command prompt**
4. Placez- vous dans le dossier du projet **dansebienP2I_VF**
5. Tapez la commande suivante **npm install** afin de télécharger les librairies dont dépend le projet.
6. Tapez la commande **expo start** pour démarrer le projet.
7. Une fois sur le Metro Bundler vous avez deux options :
 - Si vous avez un téléphone avec l'application ExpoGo :
 - Mettez le mode Tunnel et scannez le QR code avec la caméra de votre téléphone
 - Si vous avez un émulateur iOS ou Android :
 - Cliquez sur **"Run on Android device/emulator"** ou **"Run on iOS simulator"**

III. Résultat final

A. Choix techniques et architecture de l'application

Pour développer la partie front de mon application j'ai utilisé le langage JavaScript (plus précisément en Typescript), avec le framework React Native. Mon projet est composé de six principaux dossiers :

- services
- components
- screens
- navigation
- musiques
- styles

1. Services

Le dossier **services** comporte les données avec lesquelles j'ai travaillé, ainsi que les fonctions qui permettent de récupérer ces données dans d'autres pages. Pour cette application je n'ai développé que la partie front-end, c'est pourquoi les données sont inscrites en dur et sont consultables et en partie modifiables, mais il n'est pas possible d'enregistrer ses changements.

Mon projet comporte quatre classes (Cours, Exercice, Pas et Musique). Un cours correspond à une liste d'exercices. Un exercice est composé d'une liste de pas et peut être associé à une musique. On a ainsi une liste d'objets Pas dans un objet de type Exercice et une liste d'objets Exercice dans un objet de type Cours.

Les exercices sont enregistrés sans musique et ce n'est que lorsqu'ils sont dans un cours qu'on leur associe une musique.

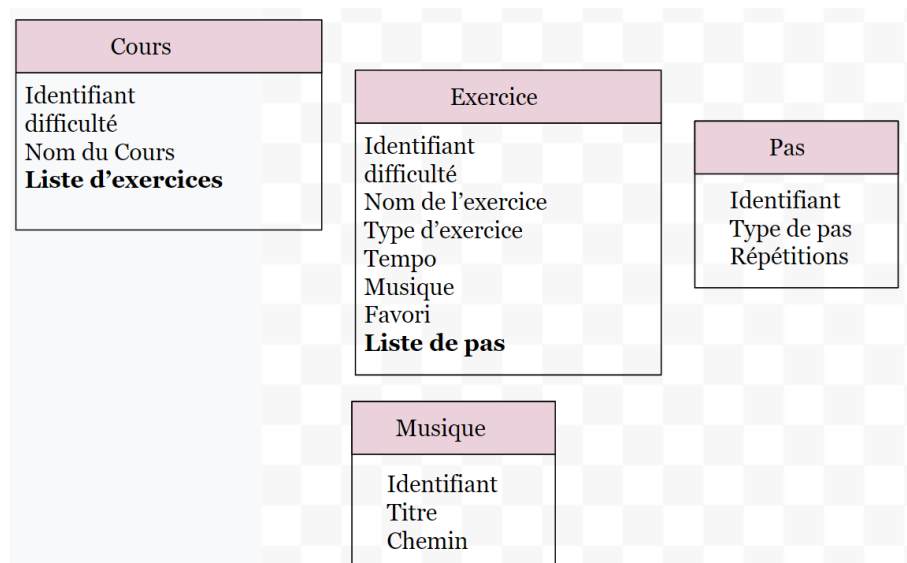


Figure 1 : résumé des quatres classes

2. Composants

Le dossier **components** contient tous les composants que j'ai utilisés pour pouvoir créer les différentes pages de l'application. Ces unités de construction propre à React ont facilité la réalisation de mon projet tenant compte de la structure de mes classes (des éléments imbriqués dans d'autres). En étant réutilisables, ces composants permettent de factoriser du code.

J'ai développé quatre types de composants, correspondant aux quatre classes de mon projet : Cours, Exercice, Pas et Musique.

	Cours	Exercice	Pas	Musique
Visualiser	CoursList CoursItem CoursDetails	ExerciceList ExerciceItem ExerciceDetails ExerciceListAjouter	PasList PasItem	MusiqueComposant
Modifier	/	ExerciceItemModifiable ExerciceDetailsModifiable	PasListModifiable PasItemModifiable	

Figure 2 : Tableau résumant les composants utilisés dans l'application

Les composants du type “...List” permettent d'afficher une liste de composants de type “...Item”. Les composants de type “...Details” permettent d'afficher les détails de chaque objet : pour un exercice cela revient à afficher ces caractéristiques ainsi que la liste de pas associée, pour un cours cela revient à afficher une liste d'exercices détaillés.

Le composant *ExerciceListAjouter* est appelé lorsqu'un utilisateur veut créer un cours et est composé d'une liste d'*ExerciceItem*. Le composant *MusiqueComposant* contient la liste des musiques enregistrées de l'application.

Pour ce qui concerne les composants de type “...Modifiable”, ils contiennent des fonctions qui permettent de modifier ou de supprimer des éléments associés : *ExerciceItemModifiable* permet d'effacer un exercice de la liste des exercices existants et dans *ExerciceDetailsModifiable* l'utilisateur modifier ou supprimer la liste de pas (en faisant appel à *PasListModifiable* et à *PasItemModifiable*).

3. Screens (écrans)

Mon projet est composé de six écrans :

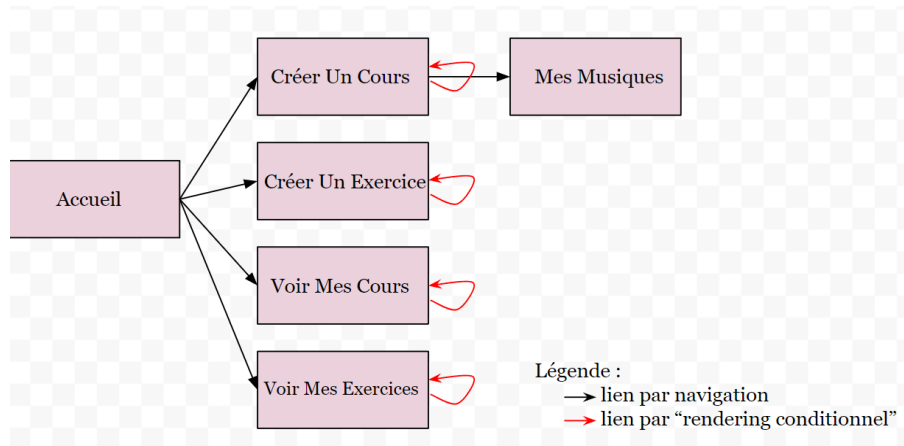


Figure 3 :diagramme faisant le lien entre les différentes pages

ACCUEIL

“Accueil” est la page principale et elle permet à l'utilisateur d'accéder aux écrans de l'application.

CRÉER UN COURS

Cette page permet à l'utilisateur de créer un nouveau cours (nommer le cours, lui donner une difficulté, et de choisir des exercices parmi ceux qui existent déjà. Une fois l'exercice choisi, l'utilisateur doit choisir une musique parmi celles qui existent déjà.

VOIR MES COURS

Cette page affiche la liste de cours, depuis cette page il est possible de consulter les cours existants qui sont composés de plusieurs exercices, dont chaque exercice est composé d'une liste de pas et d'une musique. Chaque cours est structuré de la façon suivante : *prenons l'exemple d'un cours de deux exercices, avec deux pas dans chaque exercice.*

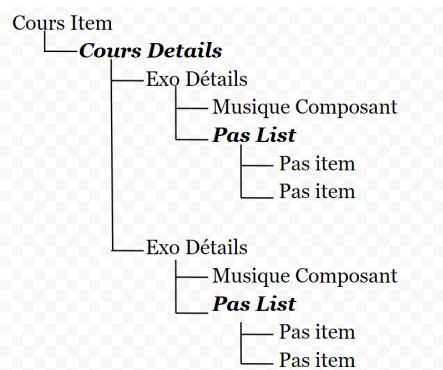


Figure 4:exemple de structure d'un cours

VOIR MES EXERCICES

Cette page affiche la liste d'exercices, depuis cette page il est possible de consulter les exercices existants qui sont composés de plusieurs pas. En plus de les consulter, il est possible de supprimer un exercice parmi cette liste, ou de le modifier en ajoutant ou en supprimant des pas. Chaque exercice est structuré de la façon suivante : *prenons l'exemple d'un exercice composé de trois pas.*

Pour le modifier, le composant PasList (et par conséquent le composant PasItem) est remplacé par le composant PasListModifiable qui comporte les fonctionnalités de suppression et de modification.

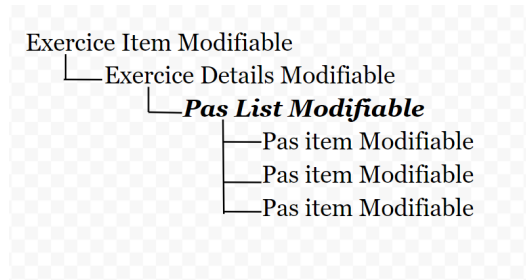


Figure 5 :exemple de structure d'un exercice

MES MUSIQUES

Cette page affiche la liste des musiques qui sont enregistrées dans le dossier musiques de l'app. Elles sont en format mp3.

4. Navigation

Le dossier **navigation** est composé de trois fichiers :

- *App.navigator.tsx* qui gère la navigation entre les pages
- *CustomNavigationBar.tsx* et *Header.tsx* qui stylise la barre de navigation, en ajoutant le bouton pour le retour en arrière

J'ai réalisé deux choix techniques pour ce qui concerne la navigation entre les écrans ([cf Figure 3](#)). D'une part j'ai utilisé la navigation appelée "stack". Pour ce type de navigation, il est possible de revenir en arrière entre les écrans vers la page d'Accueil. D'autre part, j'ai utilisé le "rendering conditionnel" pour rendre dynamique les pages. Ce système fonctionne sur la base d'un booléen qui permet d'afficher ou pas certains composants. Ainsi, j'ai créé un bouton "retour" qui modifie le booléen pour revenir à l'affichage précédent ([cf Annexe 1](#) pour un exemple de code)

5. Musiques

Le dossier **Musiques** est composé de fichiers mp3 qui correspondent aux musiques utilisées dans la création des cours.

Pour ce qui concerne la gestion du son, j'ai utilisé les dépendances [Expo-AV](#) ce qui m'a permis de manipuler des fichiers mp3 et de développer des fonctionnalités pour gérer le son.




Fonction	Bouton associé	Description
initSound	/	Création de l'objet son, pour avoir accès aux paramètres suivants : <ul style="list-style-type: none">- isPlaying- isLooping- durationMillis- positionMillis- shouldPlay
playSound		Fonction qui permet de démarrer la musique
pauseSound		Fonction qui permet de mettre la musique en pause
LoopSound		Fonction qui permet de mettre en boucle la musique
onChange	curseur	Fonction qui permet de modifier à quel instant la musique démarre
getProgress	barre colorée	Fonction qui permet d'avoir une valeur qui représente la durée en cours d'une musique pour pouvoir la représenter avec une barre qui avance

Figure 6 :Tableau résumant les fonctionnalités liées au son

La fonctionnalité EF_11 (Un utilisateur doit pouvoir accélérer ou ralentir la musique) défini en début de projet n'a pas pu être développée car les fonctions liées à la vitesse de la musique n'existent pas dans les dépendances d'EXPO-AV.

6. Styles

Le dossier **styles** est composé de trois fichiers :

- styleItem.ts
- styleTitre.ts
- styleContainer.ts

Chaque fichier contient des éléments liés au rendu visuel de l'application.

En plus des styles que j'ai développés dans ce dossier, j'ai utilisé les dépendances de *react-native-paper* qui proposent des composants avec des styles prédéfinis, pour faciliter le design de l'application.

B. Bilan des exigences

Parmi les 10 exigences fonctionnelles principales définies en début de projet, 7 d'entre elles ont été remplies. Toutes les fonctionnalités générales et les fonctionnalités liées aux exercices ont été développées, et celles liées aux musiques l'ont été partiellement ([cf Annexe 2](#)). Aucune des exigences facultatives n'a été complétée.

IV. Pistes d'amélioration

Plusieurs pistes d'amélioration se dégagent à la fin de ce projet. Tout d'abord, il serait intéressant d'associer mon application à une base de données pour pouvoir enregistrer les exercices et les cours.

Dans cette même dynamique, il serait intéressant de développer des fonctionnalités liées à la gestion des musiques, avec lesquelles les utilisateurs puissent télécharger des musiques de leurs téléphones ou lier l'application à des plateformes de streaming telles que YouTube ou Spotify ou avec une API. D'autre part, le code de plusieurs composants sont assez similaires. Il aurait été intéressant de les combiner et de développer d'autres fonctions pour éviter la répétition de code.

Finalement les exigences fonctionnelles facultatives peuvent servir de pistes d'amélioration, dans la mesure où elles visent à développer un système de compte et d'identifiants, afin de créer un réseau dans lequel professeurs et élèves puissent se partager des cours et des exercices.

V. Gestion de projet

Concernant la gestion de projet, la planification au début du projet a été réajustée à mi-projet : au début il était prévu de développer en parallèle les fonctionnalités liées aux exercices et celles liées aux musiques, mais suite à mon avancement, il a été décidé de coder la partie exercice et cours et ensuite la partie musique. Ainsi, cette nouvelle planification ([cf Annexe 3](#)) a été très utile et globalement bien respectée. Du retard a été pris notamment pendant la phase d'apprentissage du langage mais celui-ci n'a pas affecté l'aboutissement du projet.

Les trois exigences fonctionnelles non abouties sont liées à la gestion du son et plus particulièrement à la nature de l'outil que j'ai utilisé (la librairie EXPO AV). Il aurait fallu que je prévois le risque de travailler avec des dépendances qui ne proposent pas les fonctionnalités dont j'ai besoin.

Le tuteur a mis en place un système de rendez-vous bimensuel, en restant disponible selon les besoins précis par mail.

VI. Conclusion & Bilan personnel

Mon application répond globalement au Cahier des charges initial. J'ai rencontré plusieurs difficultés liées notamment à la prise en main de la technologie et la planification en début de projet mais j'ai su les gérer ce qui m'a permis d'aboutir à un résultat très satisfaisant.

Ce projet m'a permis de monter en compétences dans un langage que je ne connaissais pas avant, et ce dans un cadre de travail individuel, ce qui m'a permis de développer des méthodes de travail différentes à celles que j'avais développé à l'école dans des travaux en groupe.

Le fait que ce projet ait été réalisé sur la base d'une frustration réelle a permis de me motiver tout le long du processus. Je suis personnellement des cours de danse classique et je suis constamment face au problème de gestion de la musique : entre chaque exercice, ma professeure de danse prend énormément de temps à trouver la musique correspondante, ce qui rend peu fluide le cours. Ce projet répond ainsi à un besoin personnel et réel, ce qui rend le résultat d'autant plus satisfaisant.

VII. Annexes

Annexe 1 - Extrait du code illustrant le “rendering conditionnel” (dans écran VoirMesCours)

```
//Bouton de retour
changerR = () => {
  this.setState({ displayCoursDetails: false });
};

render() {
  if (this.state.displayCoursDetails)
    return (
      <SafeAreaView>
        <CoursDetails cours={this.state.itemCours} />

        <Button onPress={this.changerR}>Retour</Button>
      </SafeAreaView>
    );
  return (
    <View>
      <Title style={styleTitre.titre}> Mes cours</Title>

      <FlatList<Cours>
        data={this.state.cours}
        keyExtractor={(item) => item.nomduCours}
        renderItem={({ item }: { item: Cours }) => (
          <TouchableOpacity
            onPress={() => this.changerC(item)}
            style={styleContainer.mainmodifier_container}
          >
            <CoursItem cours={item} />
          </TouchableOpacity>
        )}
      />
    </View>
  );
};
```

Annexe 2 - Tableau des exigences fonctionnelles présent dans le Cahier des Charges

ID	Description de l'exigence fonctionnelle	OUI/NON
Fonctionnalités générales		
EF_1	Un utilisateur doit pouvoir organiser ces exercices selon le niveau de difficulté (si c'est un exercice pour un cours de débutant, intermédiaire ou avancé)	OUI
EF_2	Un utilisateur doit pouvoir visualiser la liste des différents exercices	OUI
EF_3	L'application doit permettre aux utilisateurs de créer un compte et de se connecter avec leurs identifiants	NON
EF_4	Un utilisateur doit pouvoir partager l'exercice et la musique à ses élèves	NON
Fonctionnalités liées aux exercices		
EF_5	Un utilisateur doit pouvoir enregistrer une combinaison de pas (un exercice)	OUI
EF_6	Un utilisateur doit pouvoir modifier et supprimer un exercice	OUI
EF_7	Un utilisateur doit pouvoir marquer comme favori un exercice	OUI
Fonctionnalités liées aux musiques		
EF_8	Un utilisateur doit pouvoir télécharger et enregistrer une musique associé à un exercice	NON
EF_9	Un utilisateur doit pouvoir remplacer ou supprimer une musique associée à un exercice	NON
EF_10	Un utilisateur doit pouvoir mettre en boucle une musique	OUI
EF_11	Un utilisateur doit pouvoir accélérer ou ralentir la musique	NON
EF_12	Un utilisateur doit pouvoir faire démarrer la musique à un moment souhaité	OUI
EF_13	L'application doit pouvoir proposer des musiques selon le type d'exercice ou la musique enregistré	NON

Annexe 3 -Planification (en tenant compte des réajustements de mi-projet) avec le temps approximatif par tâche et le temps réel dédié.

Temps approximatif	Description	Temps réel
3- 4 semaines	Prise en main de la technologie	6 semaines
1 semaine	Modélisation de la BDD et maquettage papier	1 semaine
1 semaine	Affichage exercices et filtre selon difficultés (EF 1 , EF 2)	2 semaines
1 semaine	Gérer navigation données d'un écran à un autre	1 semaine
1 semaine	Ajouter, modifier, marquer comme favori, supprimer un exercice (EF 5 , EF 6 , EF 7)	1 semaine
1 semaine	Création de cours, sur le même principe que création d'exercices	1 semaine
1 semaine	Fonctionnalités liées à la musique (EF 10 , EF 12)	2 semaines
1 semaine	Commenter le code, rédaction rapport, relecture	<1 semaine

Annexe 4 - Captures d'écran de l'interface

