# Travail Final ( 15 % ) – Cadeaux de Noël

Noël arrive à grands pas…on veut gâter nos êtres chers sans vider notre tirelire. De plus, les possibilités sont infinies et parfois on se laisse tenter d’offrir un cadeau qui finira par nous revenir par ricochet ( billets de hockey n’est pas une sortie de couple, comment ça ?? )

Nous coderons une application React Native tentant de lister nos choix de cadeaux en faisant une première tournée en ligne ou dans les magasins et ensuite réfléchir afin de faire les meilleurs choix.

## Buts du travail – seront évalués dans le travail et l’examen

* La navigation entre plusieurs écrans avec expo-router
* Le regroupement du StyleSheet dans son propre fichier et son utilisation dans les pages, incluant une bonne compréhension du module flex
* Les hooks useState
* Les FlatLists
* L’utilisation d’un composant externe à ReactNative : le slider

## Élément un peu plus difficile – expliqué dans le travail, pas à l’examen

* Le hook useContext afin de pouvoir conserver les données d’une page à l’autre ( eh oui encore un Context comme en Android Studio ! )

## Démarche suggérée : Vous démarrez avec le projet ModeleDepart qu’on a utilisé dans les différentes annexes. Npm install expo puis npx expo start comme à l’habitude.

## 1.Établir les écrans voulus et la navigation entre les écrans  :

3 écrans seront suffisants pour cette application :

* Un écran d’accueil menant aux deux autres écrans incluant une image festive
* Un écran permettant de colliger des informations sur un choix de cadeau et de les enregistrer le long de la session
* Un écran listant tous les choix de cadeaux sauvegarder à ce jour dans la session

Utilisez le composant Link ou l’objet router et établissez des routes dans le fichier \_layout.tsx de manière à pouvoir naviguer d’un écran à l’autre sans problème. Commencez par placer peu de stock dans chacun des écrans d’abord comme on a fait en classe.

« Jazzer » un peu l’apparence de vos boutons / TouchableOpacity / Pressables à l’aide d’une StyleSheet global.



*Par exemple…*

## 2. L’écran pour y ajouter un choix de cadeau.

Cet écran permettra d’enregistrer un choix de cadeau possible pour un être cher. Les informations que je voudrais que vous colligier :

* À qui se destinerait ce cadeau ( ma copine, un ami, Nicole, etc.. ) 🡪 un TextInput
* Quel est ce cadeau possible ? 🡪 un TextInput
* Quel est ( de 1 à 5 ) le prix de ce cadeau ( 1 : assez abordable, 5 : très cher ) 🡪 un Slider, composant de la librairie react-native-community
* Quel est ( de 1 à 5 ) mon opinion sur la désirabilité de ce cadeau ( 1; Nicole n’aimerait pas ce cadeau, 5 : Nicole adorerait ça ) 🡪 un Slider

Le slider à utiliser est installable avec les infos sur cette page :

<https://docs.expo.dev/versions/latest/sdk/slider/>

Lex exemples sont à cette page :

<https://github.com/callstack/react-native-slider>

Utilisation de hooks useState :

* Un <Text> doit être présent pour chaque slider, permettant d’afficher le nombre de 1 à 5 évalué dans le slider dynamtiquement 🡪 useState et utilisation du prop onValueChange dans le slider.
* Un useState devra également être établi pour les champs texte de manière à pouvoir récupérer les données pour les sauvegarder d’abord dans un tableau 🡪useState et utilisation du prop onChangeText dans les TextInput
* Un dernier useState représentant un tableau sera nécessaire. Ce dernier contiendra des objets représentant les différentes données accumulée pour un cadeau possible ( destinataire, nom du cadeau, appréciation sur 5 et prix sur 5 ). Vous pouvez ajouter un id basé sur le lenght du tableau également.

Suite à un clic de bouton, vous créerez un objet puis vous l’insérerez dans le tableau à l’aide d’une syntaxe spéciale :

setTableau([...tableau, nouvelObjet]);

Explications ( en classe ) :

setTableau :

tableau :

nouvelObjet :

…tableau :

* Utilisez ce tableau pour représenter les objets dans une FlatList à l’intérieur de votre écran actuel; ce ne sera pas le rendu final mais simplement pour vous réassurer sur le bon fonctionnement de votre useState.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*fin de la matière pour l’examen\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## 3. Ecran avec la liste des suggestions de jouets

1. Cette dernière partie du travail est un peu plus ardue mais n’ayez crainte je vais vous guider.

Bien que notre application ne sera pas pleinement persistante ( on a pas recours à une BD ou à un ficher JSON pour conserver nos données ), on veut au moins qu’on puisse afficher la même liste de suggestions de cadeaux qu’on parte de l’écran initial ou de l’écran d’ajout.

On va utiliser le contexte de notre application de manière à ce que le tableau d’objets Cadeaux ( le tableau de la page 3 ) soit accessible où qu’on soit.

On devra apprendre un nouveau hook, le useContext.

Dans son propre fichier .js, présent dans assets, codez le code suivant :

import React, { createContext, useState } from 'react';

// Créer le contexte

export const TableauContext = createContext();

// Fournisseur du contexte

export const TableauProvider = ({ children }) => {

  const [tableau, setTableau] = useState([]); // Le tableau partagé

  return (

    <TableauContext.Provider value={{ tableau, setTableau }}>

      {children}

    </TableauContext.Provider>

  );

};

2 composants sont créés :

* TableauContext : c’est un contexte qui est créé à l’aide de la méthode createContext de react, il permet de partager des données entre différents composants. Pour nous, il permettra de partager le tableau de d’objets suggestions de cadeaux entre les deux ou trois écrans
* TableauProvider : c’est le distributeur des données qui pourra partager le useState tableau ( initialement vide ) sous forme d’objet ( remarquer les {} plutôt que les [ ] dans le prop value à l’aide de la méthode useContext() dans les écrans.

1. On doit entourer notre plan de navigation ( présent dans \_layout.tsx ) du composant TableauProvider pour indiquer toutes les pages qui seront sous le contexte et qui pourront donc l’utiliser

* Envelopper votre hiérarchie de pages dans \_layout.tsx d’un composant <TableauProvider> et </TableauProvider>. Vous devez bien entendu importer le composant.

1. À présent, dans vote fichier/écran du #2, remplacer votre useState de votre tableau d’objets par useContext(TableauContext)où vous avez importé useContext de react et TableauContext de votre fichier créé au #3A.

Cependant useContext, contrairement à useState, ne retourne pas un tableau mais un objet contenant le tableau et le setTableau ( voir capture d’écran de la page précédente, sous value.

Donc, par exemple votre useState du #2 contenant le tableau de suggestion aura l’air de :

const {tableau, setTableau}= useContext(TableauContext);

Si votre #2 fonctionnait bien, rien ne devrait avoir changer pour le #3

1. Finalement, dans le 3 ème écran, vous pourrez récupérer le tableau de données de la même façon et y afficher les différents objets présents à l’aide d’un composant FlatList. Vous pourrez retirer votre FlatList du #2 une fois cette partie réussie.
2. En utilisant les flèches de navigation ainsi que des boutons, vous devriez toujours retrouver / pouvoir ajouter des idées-cadeaux et les voir dans le 3e écran.

IMPORTANT

* Remise 22 décembre minuit ou avant
* Assurez-vous d’avoir fait les #1 et #2 avant l’examen de jeudi le 19 décembre 15h20
* Assurez-vous de savoir utiliser React Native à l’école
* Étant donné l’ajout du composant slider, quand vous récupérez votre projet suite à un clone de gitHub vous devez faire :
  + npm install
  + npm install expo
  + npx expo start comme à l’habitude