**Les codes**

## 1 Provider de login

Creation du provider AuthProvider pour l'authentification

```
import React, { createContext, useState, useEffect, ReactNode } from 'react';


const initialValues = {
    email: "",
    token: "",
    authorities: "",
  };
export const AuthContext = createContext();

export const AuthProvider = ({ children }) => {
  const [user, setUser] = useState(initialValues);

  return (
    <AuthContext.Provider value={{ user, setUser }}>
      {children}
    </AuthContext.Provider>
  );
};
```

## 2 Le composant login pour l'authentification

```
import React, { useEffect } from 'react';
import {Formik, Field, Form, ErrorMessage} from 'formik';
import * as Yup from 'yup';
import "bootstrap/dist/css/bootstrap.css";
import { AuthContext } from '@/context/AuthProvider';




const validationSchema = Yup.object().shape({


    email: Yup.string()
        .email("email invalide")
        .required("l'email est obligatoire"),
    password: Yup.string()
        .required("Mot de passe est obligatoire")
        .min(4, "Mot de passe doit être plus grand que 8 caractères")
        .max(50, "Mot de passe doit être plus petit que 50 caractères"),
});
```

```
interface DetailsProps {
    cours: {
        id:number,
    },
  }

  const Login =  ({ navigation }) => {
    const initialValues = {
        email: "",
        password: "",
    };

    const { user , setUser } = React.useContext(AuthContext);

const handleSubmit = async (values) => {
    //console.log(cours.id);


    try {
        const response = await fetch('http://localhost:8080/api/login', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
            },
            body: JSON.stringify(values),
        });

        if (!response.ok) {
          //  throw new Error('Une erreur est survenue lors de l\'enregistrement des donné
            alert("mot de passe ou email incorrect");
        }

        const data = await response.json();
        console.log('Données enregistrées:', data);
        setUser(data);
        if(user){
            navigation.navigate('cours');
        }
    }

    catch (error) {
        console.error('Erreur:', error);
        // Optionally handle errors appropriately, e.g., show a notification
    }
}
```

```
useEffect (() => {
    console.log(user);

}, [user]);

    return (
        <div className="container">
            <div className="row">
                <div className="col-md-6 offset-md-3 pt-3">
                    <Formik
                        initialValues={initialValues}
                        validationSchema={validationSchema}
                        onSubmit={handleSubmit}
                        >
                        {({ resetForm }) => (
                            <Form>

                                <div className="form-group mb-3">
                                    <label htmlFor="email">
                                        email:
                                    </label>
                                    <Field
                                        type="email"
                                        id="email"
                                        name="email"
                                        className="form-control"
                                    />
                                    <ErrorMessage
                                        name="email"
                                        component="small"
                                        className="text-danger"
                                    />
                                </div>

                                <div className="form-group mb-3">
                                    <label htmlFor="password">
                                        Mot de passe:
                                    </label>
                                    <Field
                                        type="password"
                                        id="password"
                                        name="password"
                                        className="form-control"
                                    />
                                    <ErrorMessage
                                        name="password"
                                        component="small"
                                        className="text-danger"
                                    />
                                </div>
                                <div className="form-group d-flex justify-content-end gap-
                                    <button
                                        type="submit"
                                        className="btn btn-primary"
                                    >
                                        Se connecter
                                    </button>
                                    <button
                                        type="button"
                                        onClick={resetForm}
                                        className="btn btn-danger"
                                    >
                                        Annuler
                                    </button>
                                </div>
                            </Form>
                        )}
                    </Formik>
                </div>
            </div>
        </div>
    );
};
export default Login;
```

## 3 Register

```
import React from 'react';
import {Formik, Field, Form, ErrorMessage} from 'formik';
import * as Yup from 'yup';
import "bootstrap/dist/css/bootstrap.css";
import {  useRouter, } from 'expo-router';  // Importez useRouter
const router = useRouter();
const validationSchema = Yup.object().shape({
    nom: Yup.string()
        .min(3, "trop petit")
        .max(255, "trop long!")
        .required("Ce champ est obligatoire"),
    prenom: Yup.string()
        .min(3, "trop petit")
        .max(255, "trop long!")
        .required("Ce champ est obligatoire"),

    sexe: Yup.string()
    .min(2, "trop petit")
    .max(10, "trop long!")
    .required("Ce champ est obligatoire"),
    role: Yup.string()
    .min(2, "trop petit")
    .max(10, "trop long!")
    .required("Ce champ est obligatoire"),

    email: Yup.string()
        .email("email invalide")
        .required("l'email est obligatoire"),
    password: Yup.string()
        .required("Mot de passe est obligatoire")
        .min(8, "Mot de passe doit être plus grand que 8 caractères")
        .max(50, "Mot de passe doit être plus petit que 50 caractères"),
});




interface DetailsProps {
  cours: {
      id:number,



  },
}
const Register = ({ navigation }) => {
  const initialValues = {
    nom: "",
    prenom: "",
    email: "",
    sexe: "",
    role: "user",

    password: "",
};

        const handleSubmit = async (values) => {
            try {
                const response = await fetch('http://localhost:8080/api/users', {
                    method: 'POST',
                    headers: {
                        'Content-Type': 'application/json',
                    },
                    body: JSON.stringify(values),
                });

                if (!response.ok) {
                    throw new Error('Une erreur est survenue lors de l\'enregistrement des
                }
                const data = await response.json();
                console.log('Données enregistrées:', data);
               if(data){
                navigation.navigate('login');
            }

            }

            catch (error) {
                console.error('Erreur:', error);

            }
        }

    return (
        <div className="container">
            <div className="row">
                <div className="col-md-6 offset-md-3 pt-3">
                    <h1 className="text-center">Créer un compte</h1>
                    <Formik
                        initialValues={initialValues}
                        validationSchema={validationSchema}
                        onSubmit={(values) =>handleSubmit(values)}
                    >
                        {({ resetForm }) => (
                            <Form>
                                <div className="form-group mb-3">
                                    <label htmlFor="nom">
                                        nom:
                                    </label>
                                    <Field
                                        type="text"
                                        id="nom"
                                        name="nom"
                                        className="form-control"
                                    />
                                    <ErrorMessage
                                        name="nom"
                                        component="small"
                                        className="text-danger"
                                    />
                                </div>
                                <div className="form-group mb-3">
                                    <label htmlFor="prenom">
                                        prenom:
                                    </label>
                                    <Field
                                        type="text"
                                        id="prenom"
            name="prenom"
            className="form-control"
        />
        <ErrorMessage
            name="prenom"
            component="small"
            className="text-danger"
        />
    </div>

    <div className="form-group mb-3">
        <label htmlFor="email">
            email:
        </label>
        <Field
            type="email"
            id="email"
            name="email"
            className="form-control"
        />
        <ErrorMessage
            name="email"
            component="small"
            className="text-danger"
        />
    </div>
    <div className="form-group mb-3">
            <label htmlFor="sexe">Sexe:</label>
            <div role="group" aria-labelledby="sexe">
                <label>
                    <Field type="radio" name="sexe" value="mas
                    Masculin
                </label>
                <label>
                    <Field type="radio" name="sexe" value="fem
                    Féminin
                </label>
            </div><ErrorMessage name="sexe" component="small"
    </div>
    <div className="form-group mb-3">
        <label htmlFor="password">
            Mot de passe:
        </label>
        <Field
            type="password"
            id="password"
            name="password"
            className="form-control"
        />
        <ErrorMessage
```

```
                                        name="password"
                                        component="small"
                                        className="text-danger"
                                    />
                                </div>


                                <div className="form-group d-flex justify-content-end gap-
                                    <button
                                        type="submit"
                                        className="btn btn-primary"
                                    >
                                        S'inscrire
                                    </button>
                                    <button
                                        type="button"
                                        onClick={resetForm}
                                        className="btn btn-danger"
                                    >
                                        Annuler
                                    </button>
                                </div>
                            </Form>
                        )}
                    </Formik>
                </div>
            </div>
        </div>
    );
};
export default Register;
```

## 4 Liste des cours

```
import React, { useState, useEffect } from 'react';
import { View, Text, StyleSheet, FlatList, TouchableOpacity, Button } from 'react-native';
import { Link, useRouter } from 'expo-router';
import Icon from 'react-native-vector-icons/MaterialIcons';
import { AuthContext } from '@/context/AuthProvider';

const API_BASE_URL = 'http://localhost:8080/api';
const ITEMS_PER_PAGE = 5;

const ComposantCours = ({navigation}) => {
  const [cours, setCours] = useState([]);
  const [currentPage, setCurrentPage] = useState(1);
  const router = useRouter();
  const { user } = React.useContext(AuthContext);
const recupererCours = async () => {
  try {
    const response = await fetch(`${API_BASE_URL}/cours`);
    if (!response.ok) throw new Error('Erreur lors de la récupération des cours');
    const data = await response.json();
    setCours(data);
  } catch (error) {
    console.error(error);
  }
};

  console.log(user);

    if (!user) {
        alert("l'utilisateur n'est pas connecter");
        return null;
    }



useEffect(() => {
  recupererCours();
 // recupererRoleUtilisateur();  // Récupérer le rôle de l'utilisateur à chaque chargeme
}, []);

const SupprimerCours = async (id) => {
  if (user.token) {

      const response = await fetch(`http://localhost:8080/api/cours/${id}`, {
        method: 'DELETE',
        headers: {
          'Content-Type': 'application/json',
          'Authorization': `Bearer ${user.token}`,
        },
      });
      console.log(response);


      recupererCours();

  } else {
    console.log('Aucune donnée utilisateur trouvée dans localStorage.');
  }
};

const paginatedData = () => {
  const startIndex = (currentPage - 1) * ITEMS_PER_PAGE;
  return cours.slice(startIndex, startIndex + ITEMS_PER_PAGE);
};
```

```
const totalPages = Math.ceil(cours.length / ITEMS_PER_PAGE);

return (
  <View style={styles.container}>
     <Button
      title="Ajouter un Cours"
      onPress={() => navigation.navigate('addCours')}
                />
    <Text style={styles.title}>Liste des Cours</Text>

    <View style={styles.table}>
      <View style={styles.tableHeader}>
        <Text style={[styles.tableCell, { flex: 2 }]}>Titre</Text>
        <Text style={[styles.tableCell, { flex: 3 }]}>Description</Text>
        <Text style={[styles.tableCell, { flex: 1 }]}>Prix</Text>
        <Text style={[styles.tableCell, { flex: 1 }]}>Crédit</Text>
        <Text style={[styles.tableCell, { flex: 3 }]}>Actions</Text>
      </View>
      <FlatList
        data={paginatedData()}
        keyExtractor={(item) => item.id.toString()}
        renderItem={({ item }) => (
          <View style={styles.tableRow}>
            <Text style={[styles.tableCell, { flex: 2 }]}>{item.titre}</Text>
            <Text style={[styles.tableCell, { flex: 3 }]}>{item.description}</Text>
            <Text style={[styles.tableCell, { flex: 1 }]}>{item.prix} FCfa</Text>
            <Text style={[styles.tableCell, { flex: 1, paddingRight: 10 }]}>{item.credit
            <View style={[styles.tableCell, { flex: 3 }]}>
              <View style={styles.actions}>
                {/* Conditionner l'affichage des boutons 'edit' et 'delete' */}
                {/* {userRole === 'admin' && ( */}
                  <>
                    < Button
                          title="ed"
                          onPress={() => navigation.navigate('updateCours',{id:item.id,t
                    />


                    <TouchableOpacity style={styles.actionButton} activeOpacity={0.7} on
                      <Icon name="delete" size={20} color="red" />
                    </TouchableOpacity>
                    <Link href={{ pathname: '../quiz/id', params: {
                    id: item.id,
                    titre: item.titre,
                    description: item.description,
                    prix: item.prix,
                    credit: item.credit,
                  }}}>
                    <TouchableOpacity style={styles.actionButton} activeOpacity={0.7}>
```

```
                        <Icon name="quiz" size={20} color="#000" />
                      </TouchableOpacity>
                    </Link>
                    </>
                  {/* )} */}



                  {/* Bouton 'person-add' visible si connecté */}
                  {user.token && (
                    <Link href={{ pathname: '../auth'}}>
                      <TouchableOpacity style={styles.actionButton} activeOpacity={0.7}>
                        <Icon name="person-add" size={20} color="#000" />
                      </TouchableOpacity>
                    </Link>
                  )}
                </View>
              </View>
            </View>
          )}
        />
      </View>

      {/* Pagination Controls */}
      <View style={styles.pagination}>
        <TouchableOpacity onPress={() => setCurrentPage((prev) => Math.max(prev - 1, 1))}
          <Text style={styles.pageButton}>Précédent</Text>
        </TouchableOpacity>
        <Text style={styles.pageInfo}>{currentPage} / {totalPages}</Text>
        <TouchableOpacity onPress={() => setCurrentPage((prev) => Math.min(prev + 1, total
          <Text style={styles.pageButton}>Suivant</Text>
        </TouchableOpacity>
      </View>
    </View>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    padding: 20,
    backgroundColor: '#fff',
  },
  title: {
    fontSize: 24,
    fontWeight: 'bold',
    marginBottom: 20,
    textAlign: 'center',
  },
  table: {
```

```
  marginTop: 30,
},
tableHeader: {
  flexDirection: 'row',
  backgroundColor: '#f1f1f1',
  padding: 10,
  borderRadius: 8,
},
tableRow: {
  flexDirection: 'row',
  padding: 20,
  borderBottomWidth: 1,
  borderBottomColor: '#ddd',
},
tableCell: {
  flex: 1,
  textAlign: 'center',
  paddingHorizontal: 5,
  minHeight: 50,
},
actions: {
  flexDirection: 'row',
  justifyContent: 'space-around',
},
actionButton: {
  padding: 10,
  borderRadius: 5,
  alignItems: 'center',
  flexDirection: 'row',
},
buttonText: {
  marginLeft: 5,
  fontSize: 14,
},
addButton: {
  marginBottom: 10,
  padding: 10,
  backgroundColor: '#007BFF',
  color: '#fff',
  textAlign: 'center',
  borderRadius: 5,
},
pagination: {
  flexDirection: 'row',
  justifyContent: 'space-between',
  alignItems: 'center',
  marginTop: 20,
},
pageButton: {
  color: '#007BFF',
    fontWeight: 'bold',
  },
  pageInfo: {
    fontSize: 16,
  },
});

export default ComposantCours;
```

## 5 Ajouter un cours

```
import React, { useContext } from 'react';
import { Formik, Field, Form, ErrorMessage } from 'formik';
import * as Yup from 'yup';
import "bootstrap/dist/css/bootstrap.css";
import { useRouter } from 'expo-router';
import { AuthContext } from '../../context/AuthProvider'; // Corrigez le chemin d'importat

const router = useRouter();

const validationSchema = Yup.object().shape({
  titre: Yup.string()
    .min(2, "trop petit")
    .max(255, "trop long!")
    .required("Ce champ est obligatoire"),
  description: Yup.string()
    .min(2, "trop petit")
    .max(255, "trop long!")
    .required("Ce champ est obligatoire"),
  prix: Yup.number()
    .min(0, "le prix doit etre positif")
    .required("le prix est obligatoire"),
  credit: Yup.number()
    .min(0, "le credit doit etre positif")
    .required("le credit est obligatoire"),
});

const initialValues = {
  titre: "",
  description: "",
  prix: "",
  credit: "",
};

const Formulaire = () => {
  const { user } = useContext(AuthContext); // Utilisez le contexte Auth
  if (!user) {
    alert("l'utilisateur n'est pas connecté");
  return null;
}

const handleSubmit = async (values) => {
  try {
    const response = await fetch('http://localhost:8080/api/cours', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
        'Authorization': `Bearer ${user.token}`, // Utilisez le token de l'utilisateur
      },
      body: JSON.stringify(values),
    });

    if (!response.ok) {
      throw new Error('Une erreur est survenue lors de l\'enregistrement des données.');
    }

    const data = await response.json();
    console.log('Données enregistrées:', data);
    router.push('cours/liste');
  } catch (error) {
    console.error('Erreur:', error);
  }
};

return (
  <div className="container">
    <div className="row">
      <div className="col-md-6 offset-md-3 pt-3">
        <h1 className="text-center">Enregistrer un cours</h1>
        <Formik
          initialValues={initialValues}
          validationSchema={validationSchema}
          onSubmit={handleSubmit}
        >
          {({ resetForm }) => (
            <Form>
              <div className="form-group mb-3">
                <label htmlFor="titre">Titre : </label>
                <Field
                  type="text"
                  id="titre"
                  name="titre"
                  className="form-control"
                />
                <ErrorMessage
                  name="titre"
                  component="small"
                  className="text-danger"
                />
              </div>
              <div className="form-group mb-3">
                <label htmlFor="description">Description : </label>
                <Field
                  type="text"
                  id="description"
                  name="description"
                  className="form-control"
                />
                <ErrorMessage
                  name="description"
                  component="small"
                  className="text-danger"
                />
              </div>
              <div className="form-group mb-3">
                <label htmlFor="prix">Prix : </label>
                <Field
                  type="number"
                  id="prix"
                  name="prix"
                  className="form-control"
                />
                <ErrorMessage
                  name="prix"
                  component="small"
                  className="text-danger"
                />
              </div>
              <div className="form-group mb-3">
                <label htmlFor="credit"> Credit : </label>
                <Field
                  type="number"
                  id="credit"
                  name="credit"
                  className="form-control"
                />
                <ErrorMessage
                  name="credit"
                  component="small"
                  className="text-danger"
                />
              </div>
              <div className="form-group d-flex justify-content-end gap-3">
                <button
                  type="submit"
                  className="btn btn-primary"
                >
                  Enregistrer
```

```
                </button>
                <button
                  type="button"
                  onClick={resetForm}
                  className="btn btn-danger"
                >
                  Annuler
                </button>
              </div>
            </Form>
          )}
        </Formik>
      </div>
    </div>
  </div>
  );
};

export default Formulaire;
```

## 6 Modifier cours

```
import React from 'react';
import { Formik, Field, Form, ErrorMessage } from 'formik';
import * as Yup from 'yup';
import { Link, useRouter } from 'expo-router';  // Importez useRouter

import { AuthContext } from '@/context/AuthProvider';

import "bootstrap/dist/css/bootstrap.css";

const validationSchema = Yup.object().shape({
    titre: Yup.string()
        .min(2, "trop petit")
        .max(255, "trop long!")
        .required("Ce champ est obligatoire"),
    description: Yup.string()
        .min(2, "trop petit")
        .max(255, "trop long!")
        .required("Ce champ est obligatoire"),
    prix: Yup.number()
        .min(0, "le prix doit etre positif")
        .required("le prix est obligatoire"),
    credit: Yup.number()
        .min(0, "le credit doit etre positif")
        .required("le credit est obligatoire"),
});
```

```
interface DetailsProps {
    cours: {
        id:number,

        titre:string,
        description:string,
        prix:number,
        credit:number,

    },
}

const ModifierCour = ({route,navigation}) => {
    const initialValues = {
        id: route.params.id,
        titre: route.params.titre,
        description: route.params.description,
        prix: route.params.prix,
        credit: route.params.credit,
    };
    const { user } = React.useContext(AuthContext);
    console.log(user);

      if (!user) {
          alert("l’utilisateur n’est pas connecter");
          return null;
      }

    const handleSubmit = async (values) => {


        try {
            const response = await fetch(‘http://localhost:8080/api/cours/${values.id}‘, {
                method: ’PUT’, // Utilisez PUT pour mettre à jour
                headers: {
                    ’Content-Type’: ’application/json’,
                    ’Authorization’: ‘Bearer ${user.token}‘,  // Ajouter le token d’authen

                },
                body: JSON.stringify(values), // Convertit les valeurs en JSON
            });

            if (!response.ok) {
                throw new Error(’Erreur lors de la mise à jour’);
            }

            const data = await response.json();
            console.log(’Cours mis à jour:’, data);
            //router.push(’cours/liste’);
            navigation.navigate(’cours’);
    } catch (error) {
        console.error('Erreur:', error);
    }


};
return (
    <div className="container">
        <div className="row">
            <div className="col-md-6 offset-md-3 pt-3">
                <h1 className="text-center">Mettre à jour le cours</h1>
                <Formik
                    initialValues={initialValues}
                    validationSchema={validationSchema}
                    onSubmit={handleSubmit}
                >
                    {({ resetForm }) => (
                        <Form>
                            <div className="form-group mb-3">
                                <label htmlFor="titre">Titre:</label>
                                <Field
                                    type="text"
                                    id="titre"
                                    name="titre"
                                    className="form-control"
                                />
                                <ErrorMessage
                                    name="titre"
                                    component="small"
                                    className="text-danger"
                                />
                            </div>
                            <div className="form-group mb-3">
                                <label htmlFor="description">Description:</label>
                                <Field
                                    type="text"
                                    id="description"
                                    name="description"
                                    className="form-control"
                                />
                                <ErrorMessage
                                    name="description"
                                    component="small"
                                    className="text-danger"
                                />
                            </div>
                            <div className="form-group mb-3">
                                <label htmlFor="prix">Prix:</label>
                                <Field
```

```
                                        type="number"
                                        id="prix"
                                        name="prix"
                                        className="form-control"
                                    />
                                    <ErrorMessage
                                        name="prix"
                                        component="small"
                                        className="text-danger"
                                    />
                                </div>
                                <div className="form-group mb-3">
                                    <label htmlFor="credit">Credit:</label>
                                    <Field
                                        type="number"
                                        id="credit"
                                        name="credit"
                                        className="form-control"
                                    />
                                    <ErrorMessage
                                        name="credit"
                                        component="small"
                                        className="text-danger"
                                    />
                                </div>

                                <div className="form-group d-flex justify-content-end gap-
                                    <button type="submit" className="btn btn-primary">
                                        Mettre à jour
                                    </button>
                                    <button
                                        type="button"
                                        onClick={resetForm}
                                        className="btn btn-danger"
                                    >
                                        Annuler
                                    </button>
                                </div>
                            </Form>
                        )}
                    </Formik>
                </div>
            </div>
        </div>
    );
};
export default ModifierCour;
```

## 7 les navigations

```
import * as React from 'react';
import { Button, Image, StyleSheet, Text, TouchableOpacity, View } from 'react-native';
import ModifierCour from './cours/edit';
import Login from './users/login';
import ComposantCours from './cours';
import Register from './users/register';
import Formulaire from './cours/create';
// Defintion de  L'Écran d'accueil


export function HomeScreen({ navigation}) {

    return (
        <View style={styles.container}>
              <Button
                title="Se Connecter"
                onPress={() => navigation.navigate('login')}
              />
              <Text>ou</Text>
              <Button
                title="Creer Un Compte"
                onPress={() => navigation.navigate('register')}
              />
            </View>

    );
}



export function loginScreen({ navigation }) {
  return (
    <View style={styles.container}>

      <Login navigation={navigation}></Login>

    </View>
  );
}
export function registerScreen({ navigation }) {
  return (
    <View style={styles.container}>

      <Register navigation={navigation}></Register>

    </View>
  );
}
```

```
export function listeCours({ navigation }) {
  return (
    <View style={styles.container}>

      <ComposantCours navigation={navigation}></ComposantCours>
    </View>
  );
}


export function UpdateCours (props){
  const {navigation,route}=props;
  return (
  <View>

      <ModifierCour  navigation={navigation} route={route}></ModifierCour>
  </View>);


}


// Écran détails
export function CreationCoursScreen({ navigation }) {
    return (
      <View style={styles.container}>

       <Formulaire/>

        </View>
    );
  }



  export function deleteCoursScreen({ navigation }) {
    return (
      <View style={styles.container}>
        Supp
      </View>
    );
  }



  const styles=StyleSheet.create({
```

```
  container: {
     flex: 1,
     backgroundColor: '#fff',
     alignItems: 'center',
     justifyContent: 'center',
  },
  bouton:{
    padding:5,
    borderRadius:20,
    backgroundColor: '#000',
    color:'#fff',
  },
  linkButton: {
    backgroundColor: '#4CAF50', // Couleur de fond verte
    paddingVertical: 12,
    paddingHorizontal: 30,
    borderRadius: 8,
    marginVertical: 10,
    elevation: 5, // Ombre pour effet de surélévation
    shadowColor: '#000', // Ombre
    shadowOffset: { width: 0, height: 2 },
    shadowOpacity: 0.2,
    shadowRadius: 3.5,
  },
  linkText: {
    color: '#fff', // Texte blanc
    fontSize: 16,
    fontWeight: 'bold',
    textAlign: 'center',
  },
});
```