INSTITUT DE FORMATION AUX NORMES ET TECHNOLOGIES DE L'INFORMATIQUE
SOKODE

UElibre :Présentation de React native

Presenté par :SANKARA sarata & TOURE Chabane

IFNTI S4

March 3, 2025

Table des matières

1. Introduction à React native
2. Création d'un Projet React Native
3. Rôle des Fichiers et Dossiers du Projet
4. Composants
5. les composants en react
6. Les props et state
7. Les providers
8. Les Navigations
9. Gestion de l'état et des données
10. Formulaires
11. Authentification avec provider
12. Le deploiement
13. Conclusion

Introduction à React native

Introduction

Qu'est ce que react vative

React native est un framework puissant pour créer des applications mobiles multi-plateformes avec JavaScript et React.Il a été développer par facebook

Installation des Outils

Installation de l'outil Expo

- Expo est un ensemble d'outils et de services conçus pour faciliter le développement d'applications React Native.
- Pour installer Expo, utilisez les commandes suivantes:

```
npm install expo-cli -g (installation globale)
npm install expo-cli (installation locale)
```

Création d'un Projet React Native

Avec npx

Pour créer un projet React Native, utilisez les commandes suivantes.

npx create-expo-app@latest

Rôle des Fichiers et Dossiers du Projet

Voici une description des principaux fichiers et dossiers générés :
•**le dossier App** : est le coeur de notre application il contient le fichier index.tsx qui est le point d'entrée de l'application.Toutles fichiers qui sont dans ce dossier sont des navigations.
• **Le dossier assets** : Contient les images et fichiers statiques.
• **node modules** : Contient les dépendances du projet.
• **Le dossier components** : Contient les les composants du projet.
• **Le dossier constantes** : Contient les constantes du projet.
• **app.json** : Configuration de l'application. • **package.json** : Liste des dépendances et configurations.
• **Le dossier script** : contient les scripts de notre projet • **babel.config.js** : Configuration de Babel pour transpiler le code.
• **package-lock.json** : Assure la cohérence des versions des dépendances.

Démarrage et Exécution du Projet

Commandes pour démarrer le projet :

npm run start : Démarre le projet React Native.
npm run android : Démarre le projet sur Android.
npm run web : Démarre le projet sur le web

Composants

Composant

Création de composants

Un composant est une fonction ou une classe qui retourne du JSX.
Exemple :

```
function Bonjour(props) {
  return <h1>Bonjour, {props.nom}!</h1>;
}
```

Props

Utilisation des Props

Les **props** sont des arguments passés aux composants pour les personnaliser. Exemple :

```
<Bonjour nom="Jean" />
```

Composants native

Qu'est ce qu'un composant

Un composant en React Native est une unité de code représentant une partie de l'interface utilisateur. Quelques composants natives sont :

Composant Text

Le composant Text sert à afficher des chaines de texte et gère les événements tactiles.
Fourni par React Native, il est analogue à la balise ¡p¿ en HTML.

```
import React from 'react';
import {  Text,View } from 'react-native';
export default function Home() {
  return (
<View>
<Text> Bienvenue sur mon projet react native</Text>
</View>)
```

Composant Image

Le composant Image sert à afficher différents types d'images.L'attribut source prend uri, qui contient le chemin de l'image, et on peut avoir l'attribut style pour personnaliser l'affichage de l'image.

```
import React from 'react';
import {  Image } from 'react-native';
export default function Home() {
  return (
<Image style={styles.image} source={{ uri: 'https://
reactnative.dev/img/tiny_logo.png' }} />
)
```

Composant Link

Le composant Link est utilisé pour naviguer entre les écrans

```
import React from 'react';
import {  Text } from 'react-native';
import { Link } from 'expo-router';
export default function Home() {
  return (
<Link href="/propos">
<Text>a propos</Text>
</Link>)
```

Composant View

Le composant View

Conteneur prenant en charge la mise en page, le style et les contrôles d'accessibilité.

Analogue à la balise ¡div¿ en HTML

```
import React from 'react';
import { View, Text } from 'react-native';
export default function Home() {
  return (
<View>
<Text>bienvenue</Text>
</View>
);
```

Composant Stack

Le composant Stack

permer de créer une pile de navigation et permet d'ajouter de nouveaux itinéraires dans votre application

```
import { Stack} from 'expo-router';
import { View } from 'react-native';


    return (
        <View style={styles.container}>
            <Stack.Screen
              options={{
                title: "page de login",
              }}
            />
```

Composant StyleSheet

Le composant StyleSheet

Module utilisé pour créer et gérer les styles.

Avantages :

Organisation : Les styles sont centralisés.

Validation : Les propriétés des styles sont vérifiées.

Réutilisabilité : Les styles peuvent être appliqués à plusieurs composants.

Clarté : Les styles sont séparés de la logique. Les noms des propriétés suivent le format camelCase (ex. backgroundColor)

```
import  React from 'react';
import {  StyleSheet, Text, View } from 'react-native';
```

les composants en react

Nous avons deux types de composants les composants fonctionnels et les composants de classe

Les composants fonctionnel

ce sont des fonctions qui retourne du javaScript Xml (JSX) ou TypeScript xml (TSX).Exemple

```
import  React from 'react';
import {  StyleSheet, Text, View } from 'react-native';


export default function ComposantFonctionel() {
   return (

    <View style={styles.container} >
       <Text> Test ComposantFonctionel </Text>
    </View>

   );
```

Les composants de classe

ce sont des classe qui heritent de la classe Component.Exemple

```
import React, { Component } from 'react';

class Compteur extends Component {
  constructor(props) {
    super(props);
    this.state = { count: 0 }; // Déclaration du state
  }

  incrementer = () => {
    this.setState({ count: this.state.count + 1 }); // Modi
  };

  render() {
```

Les props et state

Les props

Les props

Les props : Transmettent des informations à un composant

Les state

Les state

Le state : Variable interne qui existe à l'intérieur d'un composant

Différences entre props et state

Props

Données immuables.
Transmises d'un composant à un autre.

State

Données mutables.
Utilisées uniquement à l'intérieur du composant

Les providers

Définition

Provider permet de transmettre les données du context aux composants enfants.

createContext

createContext

createContext permet de créer un contexte que les composants peuvent fournir ou lire
const SomeContext = createContext(defaultValue)

Exemple de provider 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>
  );
};
```

useContext

useContext

useContext permet de aux composant enfants d'accéder au context depuis le composant parent

Exemple

utiliser

```
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 ca
        .max(50, "Mot de passe doit être plus petit que 50
});
```

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

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

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

const handleSubmit = async (values) => {
```

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

    if (!response.ok) {
      //  throw new Error('Une erreur est survenue lors
        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., sh
    }
}

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
                                <label htmlFor="email">
                                    email:
                                </label>
```

```
                <Field
                    type="email"
                    id="email"
                    name="email"
                    className="form-co
                />
                <ErrorMessage
                    name="email"
                    component="small"
                    className="text-da
                />
            </div>

            <div className="form-group
                <label htmlFor="passwo
                    Mot de passe:
                </label>
```

```
                    <Field
                        type="password"
                        id="password"
                        name="password"
                        className="form-cor
                    />
                    <ErrorMessage
                        name="password"
                        component="small"
                        className="text-dar
                    />
                </div>


                <div className="form-group
                    <button
                        type="submit"
```

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

```
        </div>
    );
};
export default Login;
```

Les Navigations

La navigation est une bibliotheque qui permet de configurer les écrans dune application

Installation et mise en place

```
npm install @react-navigation/native
@react-navigation/native-stack

npx expo install react-native-screens
react-native-safe-area-context
```

Exemple d'utilisation

Gestion de l'état et des données

Gestion de l'état avec React

Utiliser le state local et les hooks (useState, useEffect)

le hook useState est utilisé pour déclarer des states dans un composant fonctionnel Syntaxe

```
const [state, setState] = useState(initialValue) ;
```

Exemple :

```
import React, { useState } from 'react';
function Compteur() {
// Déclare un état "count" avec une valeur initiale de 0
const [count, setCount] = useState(0);
return (
<div>
<p>Compteur : {count}</p>
<button onClick={() => setCount(count + 1)}>
Incrémenter</button>
</div>
);
}
```

useEffect est utilisé pour gérer les effets secondaires dans un composant fonctionnel. Un effet secondaire peut être toute opération qui a un impact en dehors du composant, comme la récupération de données depuis une API,syntaxe

```
useEffect(() => {
}, [dependencies]);
```

Exemple d'utilisation

Appels API et gestion des données

ntroduction à Fetch API pour récupérer des données

fecth est une méthode qui permet de contacter l'endpoint d' uneAPI et retourne une promesse Exemple :récupérer les cours depuis api de spring boot

```
const API_BASE_URL = 'http://localhost:8080/api';
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) {
```

Afficher des données dynamiques dans l'application

Pour afficher les données dynamiques nous allons utiliser le composant **FlatList**.Il a trois propriétés importantes
data : La source des données (le tableau des éléments à afficher).
keyExtractor : Fournit une clé unique pour chaque élément.
renderItem : Détermine comment chaque élément doit être affiché.
Exemple :affichons les cours récupérer de manière dynamique

```
<FlatList
data={recupererCours}
keyExtractor={(item) => item.id.toString()}
renderItem={({ item }) => (
<View style={styles.tableRow}>
<Text>{item.titre}</Text>
<Text>{item.description}</Text>
```

Formulaires

Utiliser Formik et Yup pour la validation des formulaires

Yup

Yup est une bibliothèque de validation d'objets pour JavaScript qui vous permet de définir des règles de validation pour des données, généralement dans le cadre de la gestion de formulaires.IL offre une variété de méthodes pour valider différents types de données
Exemple

```
Yup.string() : Pour valider des chaines de caractères.
Yup.number() : Pour valider des nombres.
Yup.boolean() : Pour valider des valeurs booléennes (true o
Yup.date() : Pour valider des objets de type Date.
Yup.array() : Pour valider des tableaux.
Yup.object() : Pour valider des objets.
```

Formik

Le composant Formik est utilisé pour gérer l'état du formulaire. Il prend plusieurs propriétés :
initialValues : Il définit les valeurs initiales des champs du formulaire.
validationSchema :qui vérifie si les données sont valide
onSubmit : C'est la fonction qui sera appelée lorsque le formulaire est soumis.

```
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';

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

  const handleSubmit = async (values) => {
```

```
try {
  const response = await fetch('http://localhost:8080/a
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
      'Authorization': `Bearer ${user.token}`, // Utili
    },
    body: JSON.stringify(values),
  });

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

  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<
          <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
  <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-c
      <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;
```

Conclusion

Avec ces bases, vous pouvez commencer à créer des interfaces utilisateur simples et efficaces dans React Native.