Spécification Technique pour le Composant App

1. Introduction

Le composant `App` est le point d'entrée principal d'une application React Native. Il gère la navigation entre différentes pages (Home, QR Code, NFC), maintient l'état de l'application et gère la connexion socket pour les transactions. Ce document décrit en détail les spécifications techniques du composant `App`.

2. Objectifs

- Configurer la navigation de l'application.
- Gérer la connexion et les événements socket.
- Maintenir l'état global de l'application (montant total de la transaction et clientID).

3. Dépendances

- `react`
- `react-router-native`
- `react-native`
- `TotalAmountProvider` et `useTotalAmount` pour la gestion du montant total de la transaction.
- `SocketProvider` et `useSocket` pour la gestion de la connexion socket.

4. Interface

4.1. Hooks

- `useTotalAmount`: Hook personnalisé pour accéder et mettre à jour le montant total de la transaction.
- `useSocket`: Hook personnalisé pour accéder et gérer la connexion socket.

4.2. Composants Utilisés

- `Home`: Composant pour la page d'accueil.
- `QRCode`: Composant pour la page de lecture du QR code.
- `NFCReaderPage`: Composant pour la page de lecture du NFC.

5. Fonctionnalités

5.1. Gestion de la Connexion Socket

Le composant `App` initialise et gère la connexion socket en écoutant les événements de connexion, déconnexion, et d'erreurs. Il met également à jour l'ID du client et le montant total de la transaction lorsqu'il reçoit un événement `getcheckout`.

```
useEffect(() => {
  if (!socket) return;

  socket.on('connect', () => {
    console.log(`Connected to server with socket ID: ${socket.id}`);
    socket.emit('new_tpeId', socket.id);
  });

  socket.on('connect_error', (error) => {
    console.error('Connection error:', error);
  });

  socket.on('disconnect', (reason) => {
    console.log('Disconnected from server:', reason);
  });

  socket.on('error', (error) => {
    console.error('Socket error:', error);
  });

  const handleAppStateChange = (nextAppState: AppStateStatus) => {
    if (nextAppState === 'active') {
      // Reconnect logic if needed
    } else if (nextAppState.match(/inactive|background/)) {
      // Disconnection logic if needed
    }
  };

  const subscription = AppState.addEventListener('change',
handleAppStateChange);

  socket.on('getcheckout', (data) => {
    console.log("Received 'getcheckout' event:", data.totalAmount);
    console.log("client id : " + data.clientId)
    setClientID(data.clientId);
```

```
    setTotalAmount(data.totalAmount);
  });

  return () => {
    console.log("bye");
    subscription.remove();
    socket.disconnect();
  };
}, [socket, setTotalAmount]);
```

5.2. Gestion de l'État Global

L'état global pour le montant total de la transaction et l'ID du client est maintenu à l'aide des contextes `TotalAmountContext` et `SocketContext`.

```
const { setTotalAmount } = useTotalAmount();
const socket = useSocket();
const [clientID, setClientID] = useState<string>("");
```

5.3. Navigation

Le composant utilise `NativeRouter`, `Routes`, et `Route` de `react-router-native` pour configurer la navigation entre les différentes pages de l'application.

```
return (
  <NativeRouter>
    <Routes>
      <Route path='/' element={<Home />} />
      <Route path='/QRCode' element={<QRCode clientID={clientID} />}
/>
      <Route path='/NFC' element={<NFCReaderPage
clientID={clientID}/>} />
    </Routes>
  </NativeRouter>
);
```

6. Structure du Composant

```
function App(): React.JSX.Element {
  const { setTotalAmount } = useTotalAmount();
  const socket = useSocket();
  const [clientID, setClientID] = useState<string>("");

  useEffect(() => {
    if (!socket) return;

    socket.on('connect', () => {
      console.log(`Connected to server with socket ID:
${socket.id}`);
      socket.emit('new_tpeId', socket.id);
    });

    socket.on('connect_error', (error) => {
      console.error('Connection error:', error);
    });

    socket.on('disconnect', (reason) => {
      console.log('Disconnected from server:', reason);
    });

    socket.on('error', (error) => {
      console.error('Socket error:', error);
    });

    const handleAppStateChange = (nextAppState: AppStateStatus) => {
      if (nextAppState === 'active') {
        // Reconnect logic if needed
      } else if (nextAppState.match(/inactive|background/)) {
        // Disconnection logic if needed
      }
    };

    const subscription = AppState.addEventListener('change',
handleAppStateChange);

    socket.on('getcheckout', (data) => {
      console.log("Received 'getcheckout' event:",
data.totalAmount);
```

```
      console.log("client id : " + data.clientId)
      setClientID(data.clientId);
      setTotalAmount(data.totalAmount);
    });

    return () => {
      console.log("bye");
      subscription.remove();
      socket.disconnect();
    };
  }, [socket, setTotalAmount]);

  return (
    <NativeRouter>
      <Routes>
        <Route path='/' element={<Home />} />
        <Route path='/QRCode' element={<QRCode clientID={clientID}
/>} />
        <Route path='/NFC' element={<NFCReaderPage
clientID={clientID}/>} />
      </Routes>
    </NativeRouter>
  );
}

export default () => (
  <SocketProvider>
    <TotalAmountProvider>
      <App />
    </TotalAmountProvider>
  </SocketProvider>
);
```

7. Conclusion

Le composant `App` est essentiel pour la gestion de la navigation, de l'état global et de la connexion socket dans l'application. Cette spécification technique décrit les fonctionnalités, la gestion de l'état et la navigation, ainsi que les détails de l'implémentation pour assurer une compréhension complète du composant et faciliter sa maintenance et son évolution.