

Οημιουργήνει γρηγορά μια React Native εφαρμογή προκαθορισμένες ρυθμίσεις.

4. Υποστήριξη Multi-Platform: Υποστηρίζει εφαρμογές για web, Android και iOS.

2.2 Βασικές εντολές για την υλοποίηση με React Native και Expo

Για την ανάπτυξη της εφαρμογής χρησιμοποιούμε παρακάτω βασικές εντολές του Expo:

1. npx expo init
 - Δημιουργεί μια νέα εφαρμογή React Native του Expo.
 - Παρέχει επιλογές για προκαθορισμένη αρχιτεκτονική.
2. npx expo start
 - Εκκινεί τον development server και παρέχει κωδικό για προεπισκόπηση της εφαρμογής σε Go.
3. npx expo start -c
 - Καθαρίζει την cache πριν από την εκκίνηση του development server, για επίλυση θεμάτων μνήμης.
4. npx expo install [package-name]
 - Εγκαθιστά μια βιβλιοθήκη ή πακέτο στον Expo.

3. Δομή Project

Η εφαρμογή ακολουθεί μια οργανωμένη δομή στη διαχείριση του κώδικα:

- app/: Περιέχει τις κύριες οθόνες και λειτουργίες της εφαρμογής. Π.χ. :
 - cafe.jsx, sports.jsx, shelter.jsx, weather.jsx: Για διαφορετικές ενότητες της εφαρμογής.
 - lights.jsx: Αρχείο για τη διαχείριση φωτισμού.
 - CustomProvider.jsx: Αφορά τη διαχείριση γραμματοσειρών.
- constants/: Περιέχει σταθερές τιμές ή όπως

4. Ανάλυση Κώδικα

```
import { StatusBar } from 'expo-status-bar';
```

```
native';  
import { SafeAreaView, SafeAreaProvider } from 'react-native-safe-area-context';  
import { useRouter } from 'expo-router';  
import AsyncStorage from '@react-native-async-storage/async-storage';
```

Σε όλη την εφαρμογή χρησιμοποιούνται βιβλιοθήκες όπως expo-status-bar για τον έλεγχο της μπάρας κατάστασης AsyncStorage για την αποθήκευση δεδομένων τοπικά στο συσκευή, και το useState και useEffect από React για τη διαχείριση της κατάστασης της εφαρμογής και την εκτέλεση ενεργειών κατά την εκκίνηση της εφαρμογής.

```
Αιτουηκευοή και Ανάκτηση Δεδομένων με AsyncStorage:
```

```
useEffect(() => {
  const loadSelectedButton = async () => {
    const storedButton = await
```

```
    setSelectedButton(  
    }  
};  
loadSelectedButton(  
}, []);
```

κουμπιά.

Διαδραστικά Κουμπιά και Hover Εφέ:

```
<TouchableOpacity  
    key={btn.id}
```

</TouchableOpacity>

Η χρήση του TouchableOpacity επιτρέπει την αλληλεπίδραση με τα κουμπιά. Η συνάρτηση onMouseEnter εμφανίζει πληροφορίες σχετικά με τις τοποθεσίες όταν γίνεται hover.

```
    stake.id === id ? { ...stake, [field]: value } : stake
)
);
};
```

να ενημερώσει τη συγκεκριμένη τιμή του "stake" που έχει επηρεαστεί.

εφαρμογή, όπου κάθε ομάδα μπορεί να εμφανίζεται σε ξεχωριστό row.

```
window.alert(  
    "At least one stake has an angle between 15° and 165°  
    or a pressure below 70%. \nTo continue, you should adjust the  
    angle between 0-15° & 165-180° and put the pressure over  
    75%!",  
    [{ text: "OK" }]
```

```
window.alert('Your tent has been successfully set up!  
Move on to the protective cover placement and lighting  
selection');  
    router.push("/covers");  
}  
};
```

Όταν ο χρήστης πατήσει το κουμπί για να προχωρήσει, ελέγχεται αν πληρούνται κάποιες συνθήκες για τα "stakes" (όπως αν η γωνία είναι μεταξύ 15° και 165° ή αν η πίεση είναι κάτω από το 70%). Αν δεν πληρούνται οι προϋποθέσεις,

covers.jsx:

Αλλαγή Εικόνας Σκηνής (changeTentImage):

```
const changeTentImage = (image) => {
  setTentImageSource(image);
};
```

Όταν ο χρήστης πατήσει ένα από τα κουμπιά επιλογής

καλύμματος, καλείται η συνάρτηση `changeTentImage`, η οποία αλλάζει την εικόνα της σκηνής με τη νέα εικόνα του καλύμματος.

- χρησιτής επιλεγει οιαφορετικά καλυμματά.
- lights.jsx:**
- Βιβλιοθήκες και Υποστηρικτικές Συναρτήσεις:
 - tinycolor:** Χρησιμοποιείται για να επεξεργαστεί το χρώμα (π.χ., προσαρμογή φωτεινότητας και αντίθεσης).
 - useEffect:** Ενημερώνει το χρώμα (`adjustedColor`) όταν αλλάζουν οι τιμές του `selectedColor`, `brightness` ή `contrast`.
 - useRouter:** Χρησιμοποιείται για τη διαχείριση της

```
const modifyColor = (color) => {
    return tinycolor(color)
        .lighten((brightness - 1) * 100) // Προσαρμογή
        φωτεινότητας
        .saturate((contrast - 1) * 100) // Προσαρμογή
```

```
αντίθεσης  
    .toString0;  
};
```

Σω γίνεται η εμφανισή του περιεχομένου ανάλογα με το επιλεγμένο currentScreen. Ο χρήστης μπορεί να αλλάξει οθόνες με βάση τις επιλογές του, πηγαίνοντας από την αρχική σελίδα στην οθόνη χρωματισμένων εφέ φωτισμού ή τα προετοιμασμένα εφέ φωτισμού.

Αφού ανοιξουμε το TentifyApp folder στο VSCode (ή σε όποιο άλλο IDE επιθυμουμε), ανοίγουμε το Terminal και πληκτρολογούμε “cd TentifyReactN” και αφού την εκτελέσουμε, πληκτρολογούμε την εντολή “npx expo start -c”, την οποία επίσης εκτελούμε.

Η πρώτη εντολή αλλάζει τη διαδρομή στο terminal (cd = change directory) και η δεύτερη είναι αυτή που εκκινεί τον development server (και παράλληλα καθαρίζει και την cache {-c}). Για να τερματίσει το project πατάμε Ctrl + C.

A QR code is shown on the left, with a large arrow pointing from it to the right text. Below the QR code, a list of developer commands is provided, each with a corresponding arrow pointing to its translation in Greek on the right.

- > Metro waiting on <exp://192.168.68.110:8081>
- > Scan the QR code above with Expo Go (Android) or the Camera app (ios)
- > Web is waiting on <http://localhost:8081>
- > Using [Expo Go](#)
- > Press **s** | switch to development build
- > Press **a** | open Android
- > Press **w** | open web
- > Press **t** | open debugger

QR Code ή link για άνοιγμα της εφαρμογής στο EXPO GO.

link για άνοιγμα της εφαρμογής στο web (ανοίγει και με το πάτημα του 'w').

Logs for your project will appear below. Press Ctrl+C to exit.