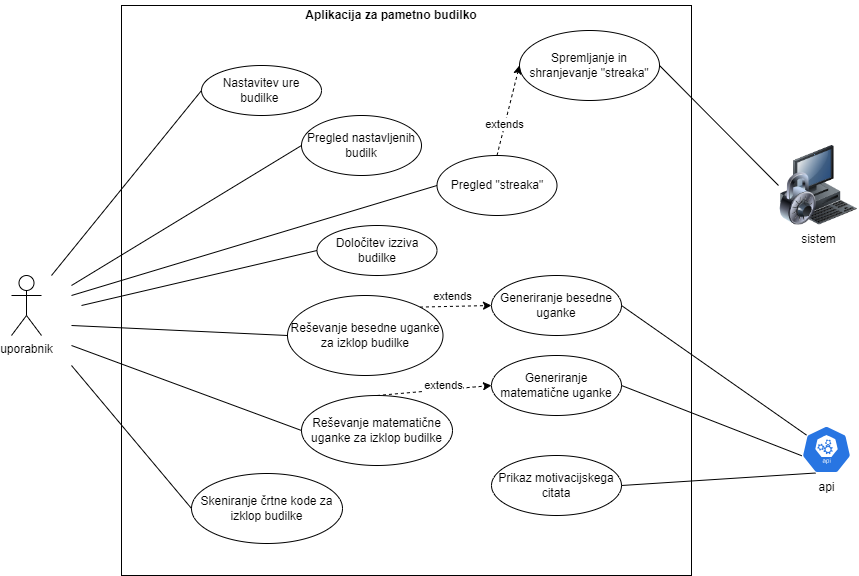
WakeMeApp - Pametna budilka

Predstavitev teme

Mobilna aplikacija WakeMeApp prebudi uporabnika z uporabo interakcije in motivacije. Za izklop zvonjenja zahteva interakcijo, kot je skeniranje barkode ali reševanje matematične enačbe, ob tem pa ponuja personalizirane spodbude za jutranjo motivacijo. Aplikacija bi ponujala tudi možnost postopnega glasnejšega zvonjenja za nežno prebujanje.

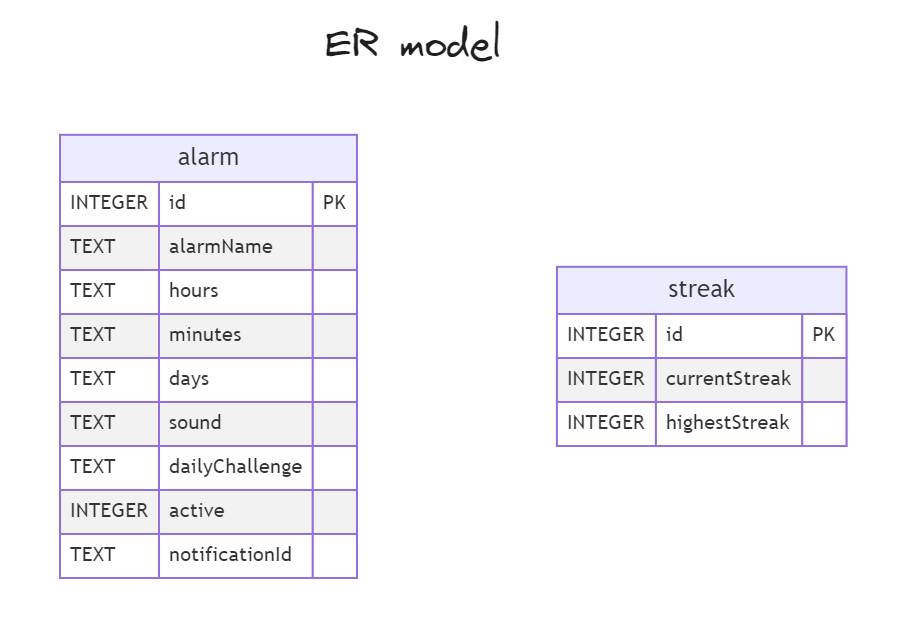
Diagram primerov uporabe



Akterji v aplikaciji so uporabniki, sam sistem ter api-ji, ki so bili uporabljeni tekom razvoja projekta za pridobivanje različnih podatkov. Uporabnikom aplikacija nudi različne funkcionalnosti kot so nastavitev in pregled budilke pa vse do reševanja zabavnih izzivov ter spremljanja svojega dnevnega napredka skozi »streak«.

Shema podatkovne baze

Za prikaz podatkovne baze je bil uporabljen ER-model.

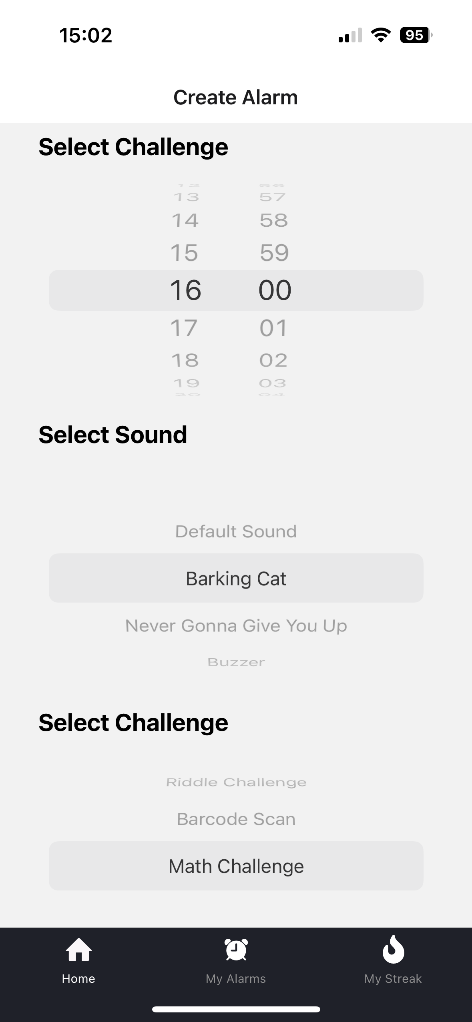


Zaslonske slike grafičnega vmesnika

V naslednjem sklopu so prikazani zaslonski posnetki grafičnega vmesnika. Za »homepage« aplikacije je nastavljena stran za dodajanje nove budilke, ki vsebuje možnosti za poimenovanje budilke, izbiro zvoka in dnevnega izziva ter nastavitev čas. S klikom na gumb »Save Alarm« se budilka shrani med ostale morebitne budilke.

A screenshot of a phone

Description automatically generated A screenshot of a phone

Description automatically generated 

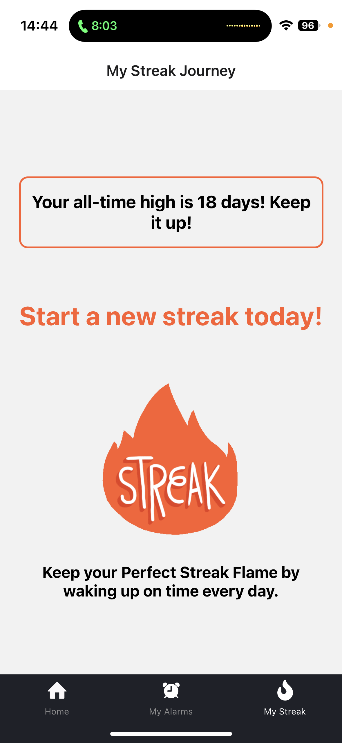
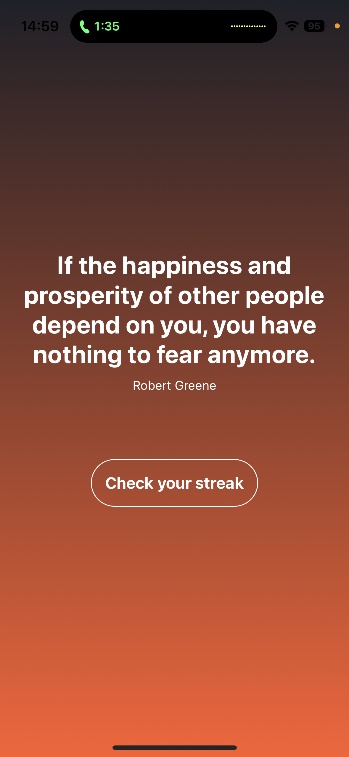
Aplikacija omogoča izbiro med kar tremi različnimi izzivi. Ti izzivi so »BarcodeChallenge«, »MathChallenge« ter »RiddleChallenge«, pri čem ima vsak od njih drugo, zanimivo nalogo. Z podajo pravilnega odgovora v določenem številu poskusov ali pred potekom časa se naloga uspešno reši. Napredek se shrani za izpis »streaka«.

A screenshot of a math test

Description automatically generated A screenshot of a quiz

Description automatically generated A screenshot of a barcode scanner

Description automatically generated

Aplikacija spodbuja uporabnike k jutrnji motivaciji z izpisom streak-a ter dnevnimi citati. Streak se dnevno povečuje z uspešnim opravljanjem naloge.

Vodič za implementacijo

1. Vodič home.tsx

**1. Uvoz knjižnic in komponent:**

Uvoziti je potrebno knjižnice in komponente, ki jih potrebujete za izgradnjo vaše aplikacije. Na primer, **react, react-native, @react-native-community/datetimepicker, @react-native-picker/picker, expo-notifications, in react-native-gesture-handler**.

**2. Nastavitev rokovanja z obvestili:**

Uporabiti je potrebno **Notification.setNotificationHandler**, da nastavite rokovanje s obvestili v vaši aplikaciji. V tem primeru se izvaja funkcija **handleNotification**, ki določa, kako naj se obnašajo obvestila (predvajanje zvoka, nastavitev odznake, prikaz opozorila).

**3. Konfiguracija obvestil:**

V funkciji **configurePushNotifications** preverite, ali ste pravilno nastavili kanal za obvestila na Androidu in pridobili dovoljenje uporabnika za prejemanje obvestil. Poskrbite, da so dovoljenja za obvestila odobrena.

**4. Uporaba useEffect:**

Uporabite **useEffect** za izvedbo konfiguracije obvestil ob začetku komponente in poslušanje dogodkov prejemanja obvestil.

**5. Določanje izziva, imena ter delo s časom in izbira zvoka:**

Izberite izziv za alarm iz možnosti, kot so "Barcode Scan", "Riddle Challenge" in "Math Challenge". Uporabite vmesnik **SelectPicker**. Vnesite ime alarma preko vnosnega polja **TextInput**. Implementirajte funkcije, ki omogočajo izbiro časa za alarm in izbiro zvoka iz seznama. Uporabite vmesnik za izbiro časa.

**6. Shranjevanje alarma:**

Ko določite vse nastavitve alarma in pritisnete gumb "Save Alarm", sprožite funkcijo **handleSaveAlarm**. Ta funkcija nastavi čas alarma, ustvari obvestilo in shrani podatke o alarmu v podatkovno bazo.

**7. Izvedba alarma v knjižnici Database:**

Dodajte obvestila o alarmih v podatkovno bazo preko knjižnice **Database**.

1. Vodič BarcodeChallenge:

**1. Uvozite potrebne knjižnice in vire zvoka:**

Prepričajte se, da ste pravilno uvozili vse potrebne knjižnice, kot so **expo-barcode-scanner, react, react-native, expo-av, expo-router** in druge. Prav tako poskrbite, da so zvočne datoteke, kot so **Buzzer.mp3, BarkingCat.mp3, RickRoll.mp3 in Default.mp3**, pravilno uvožene.

**2. Nastavitev stanj:**

Uporabite **useState** kuke za spremljanje stanj, kot so preostali čas (**remainingTime**), dovoljenje za uporabo kamere (**hasPermission**), ali je bil že zaznan črtni kod (**scanned**), ali je bil gumb za skeniranje pritisnjen (**isClicked**), in zvok (**sound**).

**3. Pridobitev dovoljenj za uporabo kamere:**

Uporabite **BarCodeScanner.requestPermissionsAsync()** za pridobitev dovoljenj za uporabo kamere.

**4. Predvajanje zvoka za zadnji alarm:**

Ob zagonu komponente uporabite **Database.getLatestAlarm** za pridobitev podatkov zadnjega alarma iz podatkovne baze. Nato predvajajte zvok tega alarma s funkcijo **playSound**. Uporabite **Audio.Sound.createAsync** za predvajanje zvoka izbrane melodije ob začetku izziva. Implementirajte funkcionalnost za ustavitev in sprostitev zvoka ob koncu izziva.

**5. Samodejno štetje časa izziva:**

Uporabite **setInterval**, da samodejno odštejete preostali čas vsako sekundo. Nastavitev **clearInterval** je uporabljena v **useEffect**, da se interval izklopi ob odstranitvi komponente.

**6. Obvladovanje skeniranja črtnega koda:**

Ko se zazna črtni kod, sprožite funkcijo **handleBarCodeScanned**, ki ustavi zvok, poveča števec izzivov v podatkovni bazi, in prikaže sporočilo o uspehu.

**7. Prikaz črtnega čitalca:**

Uporabite **BarCodeScanner** komponento za prikaz črtnega čitalca, ki se aktivira, ko uporabnik pritisne gumb "Scan Barcode".

**8. Prikaz preostalega časa in gumbov:**

Vključite prikaz preostalega časa, gumba "Scan Barcode" in gumba za ponovno skeniranje po uspešnem skeniranju.Če čas poteče, sprožite dogodek, ki uporabnika obvesti, da je čas potekel, ponastavi števec izzivov v podatkovni bazi in preusmeri uporabnika na ustrezni zavihek.

1. Vodič za StreakScreen

**1. Uvozite potrebne knjižnice in vire slik:**

Prepričajte se, da ste pravilno uvozili vse potrebne knjižnice, kot so **react, react-native, @react-navigation/native**, in druge. Prav tako poskrbite, da je slika **fire.gif** pravilno uvožena.

**2. Nastavitev stanj:**

Uporabite **useState** za spremljanje stanj, kot sta trenutni niz (**streak**) in najvišji niz (**highest**).

**3. Pridobitev podatkov o nizu:**

Implementirajte funkcijo **fetchStreakCount**, ki uporablja **Database.getStreak** za pridobitev podatkov o trenutnem in najvišjem nizu iz podatkovne baze.

**4. Uporaba useFocusEffect za obnavljanje podatkov:**

Uporabite **useFocusEffect** za zagotovitev, da se podatki o nizu posodabljajo vsakič, ko je zaslon osredotočen.

**5. Prikaz najvišjega niza:**

Uporabite komponente **<Text>** za prikaz sporočila o najvišjem nizu. Sporočilo naj se razlikuje glede na to, ali je najvišji niz večji od 0 ali ne.

**6. Prikaz trenutnega niza:**

Uporabite komponente **<Text>** za prikaz trenutnega niza. Sporočilo naj se razlikuje glede na to, ali je trenutni niz večji od 0 ali ne.

**7. Uporaba slike v ozadju:**

Uporabite **<ImageBackground>** komponento za prikaz slike v ozadju. Poskrbite, da je slika pravilno izvor zahtevana v **require**.

**8. Prikaz dodatnega besedila:**

Uporabite **<Text>** komponento za prikaz dodatnega besedila pod sliko. Prilagodite besedilo glede na želje vaše aplikacije.

**9. Stilizacija komponent:**

Prilagodite stilizacijo glede na oblikovanje vaše aplikacije. Posvetite pozornost barvam, velikostim pisav, zaobljenosti robov in drugim vizualnim elementom.