Tampereen ammattikorkeakoulu

Sovellusohjelmoinnin jatkokurssi

Oppimispäiväkirja

Antti Venetjoki

SISÄLLYS

1 Viikkotehtävät

1.1 Mobiilisovelluskehityksen yleiskuva

Natiivi mobiilikehityksessä kehitetään sovellusta tietylle alustalle tai käyttöjärjestelmälle. Nykypäivänä yleisimmät ovat iOS ja Android. Natiivikehityksessä myös yleisesti valitaan jokin käyttöjärjestelmä versio, jonka ominaisuuksia käytetään hyväksi kehityksessä. Tämä myös tarkoittaa, että laitteet, joiden käyttöjärjestelmän versio on vanhempi kuin kehityksessä valittu versio, niin sovellus ei toimi kyseisellä laitteella.

Cross-platform kehityksessä käytetään sovelluskehystä, jota käyttäen voidaan kehittää sovellus useammalle alustalle. Muodoltaan sovelluskehykset ovat kirjastoja, joista löytyy hyvin määritelty ohjelmointirajapinta (API), jonka alta löytyy implementaatiot monelle eri alustalle.

Web-kehityksessä kehitetään ohjelmia, jotka toimivat verkkoselaimen päällä. Tällöin käytännössä kaikki laitteet, joista löytyy verkko selain pystyvät käyttää ohjelmaa.

1.2 React Native

1.2.1 Pohdinta

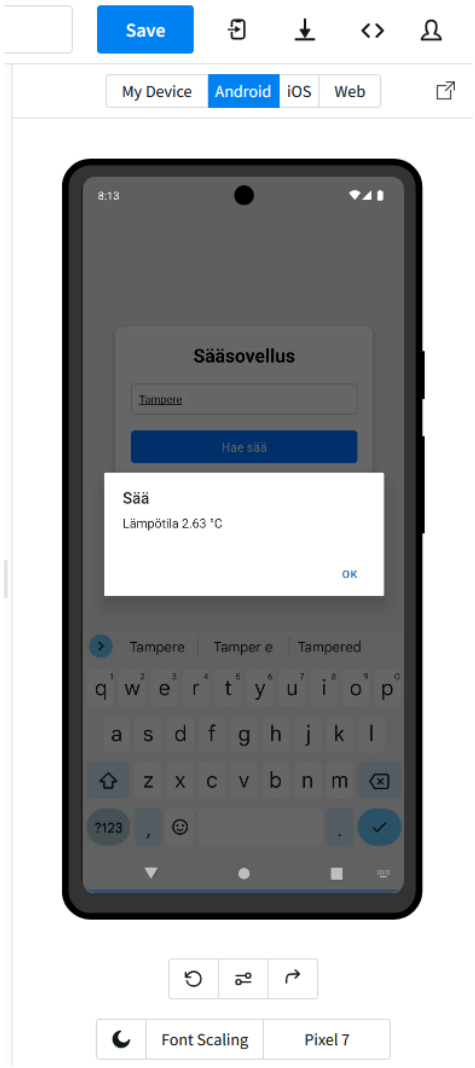
React on suunniteltu web-sovelluksille ja käyttää HTML:ää sekä CSS:ää, kun taas React Native on tarkoitettu mobiilisovelluksille (iOS ja Android), ja se käyttää natiivikomponentteja kuten <View> ja <Text>. React toimii selaimessa, kun taas React Native kääntää koodin natiiviksi mobiilisovellukseksi. Lisäksi tyylit määritellään eri tavalla: Reactissa CSS:n avulla ja React Nativessa JavaScript-objekteina.

En saanut ympäristööni toimimaan vielä, joten alla oleva ohjelma on testattu snack.expo.dev verkkoympäristössä.

1.2.2 Github-linkki

github.com

1.2.3 Todiste ohjelman toiminnasta



2 Viikkoharjoitukset 2

2.1 Pohdinta

SafeAreaView on React Nativessa komponentti, joka varmistaa, että sovelluksen sisältö ei jää piiloon laitteiden loven (notch), pyöristettyjen kulmien tai muiden esteiden alle. Se erityisesti hyödyttää moderneja laitteita, joilla on epäsymmetrisiä näyttöjä, kuten iPhone. Käyttämällä *SafeAreaView*:tä varmistat, että käyttöliittymä mukautuu laitteiden turvallisiin alueisiin.

Responsiivinen layout React Nativessa voidaan toteuttaa käyttämällä:

- **Flexboxia** asetteluun.
- **Dimensions API:** avulla saat ruudun koon ja voit mukauttaa tyylejä.
- **Prosenttipohjaiset arvot** leveyteen, korkeuteen, marginaaleihin ja fonttikokoon.
- **Skaalautuvat yksiköt:** Kirjastot kuten *react-native-responsive-screen* tarjoavat prosentuaalisia ratkaisuja.
- **Media query -tyyppinen lähestymistapa** voidaan toteuttaa kirjastoilla kuten *react-native-media-queries*.

2.2 Github-linkki

github.com

2.3 Kuva ohjelman toiminnasta

3 Viikkoharjoitukset 3

3.1 SafeAreaView

SafeAreaView on React Nativen tarjoama komponentti, joka auttaa sovellusta mukautumaan erilaisten mobiililaitteiden näytön muotoihin ja turvallisiin alueisiin. Tämä on erityisen hyödyllistä laitteissa, joissa on lovi ("notch"), pyöristetyt kulmat tai erillinen status bar (kuten iPhonet). SafeAreaView luo näytölle "turva-alueen" (safe area), joka estää sisällön näkymisen lovien, status barien tai pyöristettyjen kulmien alueilla. Näin ollen SafeAreaView parantaa sovelluksen käytettävyyttä ja ulkonäköä kaikenkokoisilla ja -muotoisilla laitteilla.

Käyttötarkoitus: SafeAreaView-komponenttia kannattaa käyttää komponenttien ympärillä, joilla on tärkeä visuaalinen tai toiminnallinen rooli, ja jotka tulisi pitää pois näytön reunoilta. Erityisesti ylä- ja alareunan elementit, kuten navigointi-palkit, saavat tällä tavoin riittävän tilan eikä niitä estetä.

3.2 Responsiivinen layout

Dimensions API: React Nativessa Dimensions-API mahdollistaa näytön leveyden ja korkeuden hakemisen, jolloin layoutin tai fonttikokojen säädöt voidaan määrittää niiden perusteella.

PixelRatio-moduuli: PixelRatio-moduulin avulla voi säätää tyylimäärityksiä, kuten fonttikokoa ja marginaaleja näytön koon perusteella.

react-native-responsive-screen-kirjasto: Tämä kirjasto sisältää käteviä työkaluja, kuten widthPercentageToDP ja heightPercentageToDP, joilla saa asetettua prosenttipohjaisia leveyksiä ja korkeuksia.

useWindowDimensions-hook: Tämä Reactin hook mahdollistaa näytön leveyden ja korkeuden dynaamisen hakemisen, jolloin layout mukautuu automaattisesti, jos laitteen suunta muuttuu.

3.3 StyleSheet-esimerkki

```
import { StyleSheet } from 'react-native';

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    padding: 16,
  },
  header: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    fontSize: 24,
    fontWeight: 'bold',
    color: '#333',
  },
  weatherInfo: {
    flex: 3,
    justifyContent: 'center',
    alignItems: 'center',
    fontSize: 18,
    color: '#666',
    padding: 10,
  },
  refreshButton: {
    flex: 1,
    justifyContent: 'flex-end',
    marginBottom: 20,
    padding: 10,
    backgroundColor: '#008CBA',
    borderRadius: 5,
  },
  buttonText: {
    fontSize: 16,
    color: '#fff',
    fontWeight: '600',
    textAlign: 'center',
  },
});

export default styles;
```

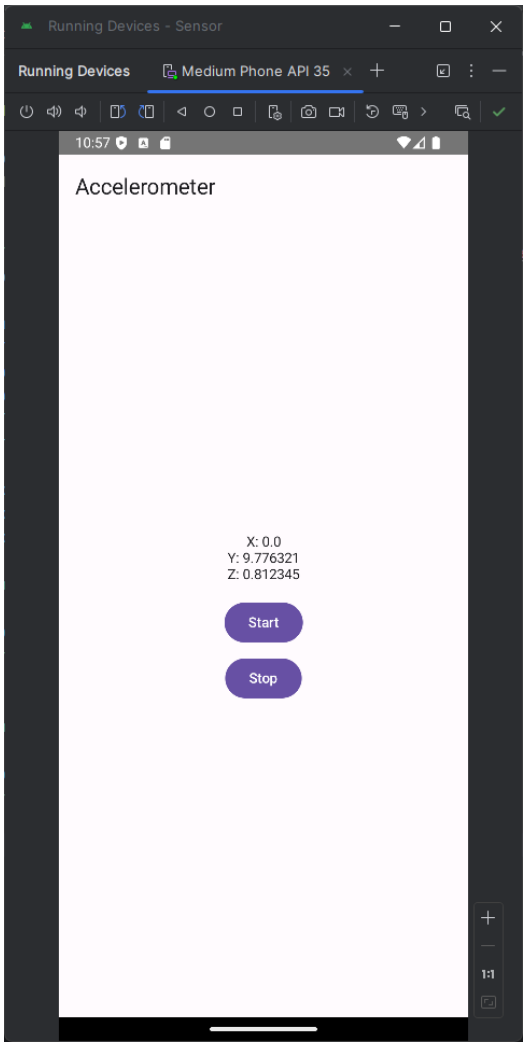
4 Viikkoharjoitukset 4

4.1 Device Managerit ja kiihtyvyysanturi

4.1.1 Mitä ovat Device Managerit

Device Managerit Android-järjestelmässä ovat ohjelmisto-rajapintoja (APIt), jotka tarjoavat sovelluksille pääsyn laitteiston ja järjestelmän hallinnoimiin resursseihin, kuten antureihin, sijaintiin, kameraan tai verkkoyhteyksiin. Ne toimivat välittäjinä sovellusten ja Android-järjestelmän välillä, tarjoten turvallisen ja yhtenäisen tavan käyttää laitteistoresursseja.

4.1.2 Kuva esimerkki ohjelmasta



4.1.3 Github-linkki

github.com

4.2 Permissiot

4.2.1 Mitä on permissiot

Permissiot eli käyttöoikeudet ovat Android-järjestelmän mekanismi, jolla hallitaan sovellusten pääsyä järjestelmän tai laitteen suojattuihin resursseihin ja käyttäjätietoihin. Niiden tarkoitus on suojata käyttäjän yksityisyyttä ja estää sovelluksia käyttämästä resursseja ilman lupaa.

Esimerkkejä resursseista, joita hallitaan permissioilla:

- Käyttäjän sijainti.
- Kamera ja mikrofonit.
- Tallennustila.
- Puhelulokit ja yhteystiedot.

4.2.2 Missä tilanteissa permissioita tarvitaan

Sovellukset tarvitsevat permissioita aina, kun ne yrittävät käyttää tietoja tai resursseja, joita pidetään käyttäjän kannalta yksityisinä tai kriittisinä. Esimerkkejä:

- **Sijainnin käyttö**: Sovellus, joka näyttää karttoja tai tarjoaa navigointipalveluja, tarvitsee pääsyn käyttäjän GPS-tietoihin.
- **Kameran käyttö**: Kameraa käyttävä sovellus (esim. valokuvaus- tai QR-koodinlukijasovellus) tarvitsee pääsyn laitteen kameraan.
- **Yhteystiedot**: Sovellus, joka lähettää viestejä tai hallinnoi kontakteja, tarvitsee pääsyn yhteystietoihin.
- **Tallennustila**: Sovellus, joka tallentaa tiedostoja (kuten valokuvia tai asiakirjoja), tarvitsee pääsyn laitteen sisäiseen tai ulkoiseen tallennustilaan.

4.2.3 Dangerous ja Non-dangerous permissiot

Permissiot jaetaan kahteen pääluokkaan: **dangerous** ja **non-dangerous**.

Dangerous permissions

- Näihin liittyy käyttäjän yksityisyys tai turvallisuusriski.
- Esimerkkejä: Sijainti, yhteystiedot, kamera, puhelulokit.
- Käyttäjän täytyy erikseen hyväksyä nämä käyttöoikeudet joko sovelluksen käytön aikana (runtime permission) tai sovelluksen asennuksen yhteydessä (vanhemmissa Android-versioissa).

Non-dangerous permissions

- Näihin ei liity käyttäjän yksityisyyteen tai turvallisuuteen liittyviä riskejä.

- Esimerkkejä: Internet-yhteyden käyttö, sovelluksen tilan tarkistaminen (READ_APP_STATE).
- Nämä käyttöoikeudet myönnetään automaattisesti, mutta ne täytyy silti määrittää sovelluksen manifest-tiedostossa.

Sovelluskehittäjän näkökulmasta ero:

- **Dangerous permissions**: Kehittäjän täytyy implementoida mekanismi pyytääkseen käyttäjältä lupaa sovelluksen käytön aikana.
- **Non-dangerous permissions**: Näitä ei tarvitse pyytää erikseen, mutta ne täytyy listata AndroidManifest.xml-tiedostoon, jotta Android-järjestelmä voi tarkistaa, että sovellus ilmoittaa käyttävänsä kyseistä resurssia.

Miksi non-dangerous permissiot tulee listata manifest-tiedostossa?

Manifest-tiedosto toimii sovelluksen määrittelynä ja kertoo Android-järjestelmälle, mitä resursseja sovellus käyttää. Jos non-dangerous permissiot jätetään manifestista pois, sovellus ei pysty hyödyntämään niitä resursseja, koska järjestelmä ei tiedä, että sovellus tarvitsee niihin pääsyä.

4.2.4 Lupien tarkistaminen

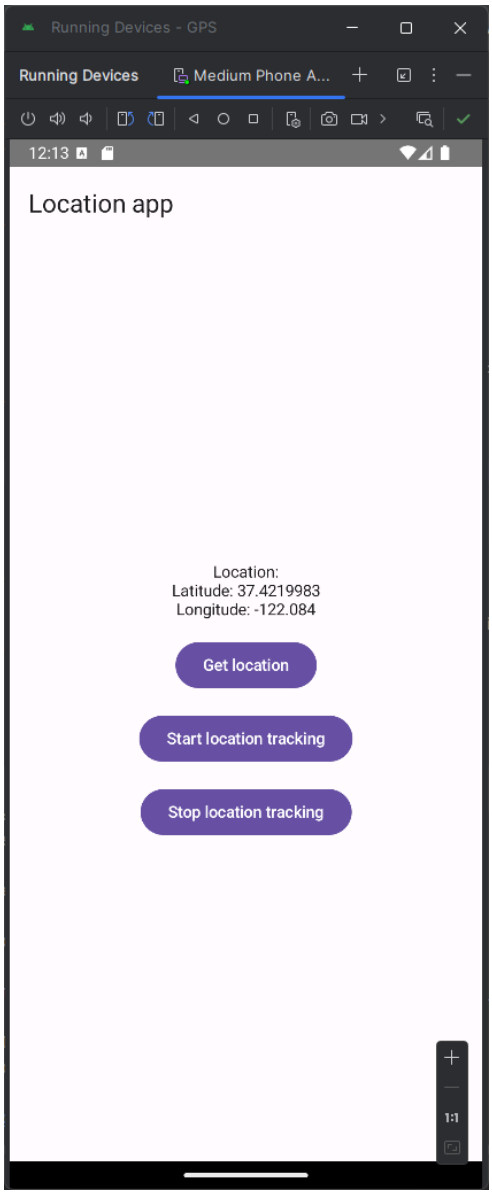
Android tarkistaa permissiot seuraavasti:

1. **Manifest-tiedosto**: Ensin tarkistetaan, onko sovelluksen manifest-tiedostossa määritelty tarvittava permission.
2. **Runtime permission (jos applicable)**: Jos resurssi kuuluu **dangerous permission** -luokkaan, Android tarkistaa, onko käyttäjä nimenomaisesti hyväksynyt käyttöoikeuden sovelluksen asennuksen tai käytön aikana.
3. **Suorituksen aikana**:
    - Android-järjestelmä vertaa resurssipyyntöä sovelluksen käyttöoikeusluetteloon.
    - Jos lupa on myönnetty, resurssin käyttö sallitaan.
    - Jos lupa puuttuu, järjestelmä hylkää pyynnön ja heittää virheen, joka voi olla esimerkiksi SecurityException.

Tässä tarkistusprosessissa Android suojaa käyttäjän yksityisyyttä estämällä luvattomat pääsyt kriittisiin tietoihin tai resursseihin.

4.3 Permission pyytäminen käyttäjältä ja paikkatiedon seuraaminen

4.3.1 Kuva esimerkki ohjelmasta



4.3.2 Github-linkki

github.com

Käytetyt lähteet

snack.expo.dev
https://reactnative.dev/
https://home.openweathermap.org/
https://github.com/facebook/watchman/tree/v2024.09.30.00
https://carbon.now.sh