**SVEUČILIŠTE U RIJECI**

**FAKULTET INFORMATIKE I DIGITALNIH TEHNOLOGIJA**

**2. SEMINARSKI RAD IZ KOLEGIJA MREŽNI I MOBILNI OPERACIJSKI SUSTAVI**

**DOKUMENTACIJA APLIKACIJE**

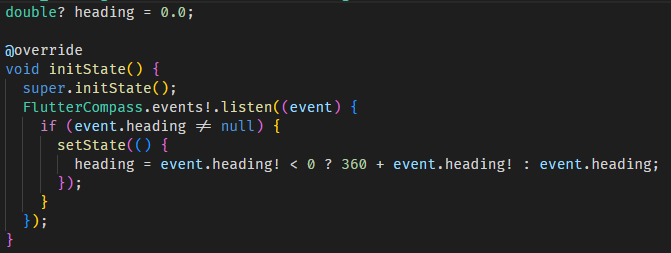
**Autor: Leo Vukoje**

**Nositelj: izv. prof. dr. sc. Božidar Kovačić**

**Asistentica: Matea Turalija**

Repozitorij: <https://github.com/Vuki0901/Flutter-Compass-App>

Za ovaj seminarski rad sam se odlučio iskoristiti senzor magnetometar kako bi izradio aplikaciju koja služi kao kompas. Na zakretanje mobitela u listener se vraća event objekt koji sadrži nekoliko svojih property-a, od kojih je jedan *heading*, koji nam govori o položaju mobitela u odnosu na sjeverni pol u stupnjevima.



FlutterCompass je klasa iz službenog paketa flutter\_compass koja pruža pristup eventu od kompasa koji vraća *heading* čija vrijednost može biti od 0 do 360, gdje 0 označava sjever.

U odlomku koda gore postavljamo *listener-a* da prati vrijednost koju kompas vraća, te da na svaku promjenu ažurira i varijablu *heading* koja označava smjer mobitela.

U Flutteru svaki *widget* koji gradimo mora imati metodu build koja vraća ono što želimo prikazati na ekranu:

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

Ovakav widget se sastoji od više manjih widgeta koji ga tvore. Tako svaki ekran u Flutteru započinje widgetom *Scaffold* kojemu se može postaviti *appBar*, *body*, *bottomNavigationBar*, *menu* i razni drugi elementi koji mogu tvoriti jedan ekran.

Unutar appBar-a se stavlja *AppBar* widget u koji mogu ići razni elementi na razna mjesta (property-ji *leading* i *trailing*). Ovdje sam ja stavio samo naslov sa tekstualnim stilom.

U body ide glavni sadržaj aplikacije, ovdje sam ja stavio da svi elementi idu u jedan *Column* (stupac) iako postoje brojni drugi načini za raspoređivanje sadržaja, *SingleChildScrollView* ako će sadržaj premašiti duljinu ekrana, *GridView* ako želimo rasporediti sadržaj unutar željene rešetke i dr. Unutar

*Column-a* sadržaj se reda jedan ispod drugog, tako su ovdje 3 glavna widgeta u *Colmun-u*, prvi je *Text* koji služi za prikaz varijable *heading*, drugi je *SizedBox* koji služi samo za stvaranje razmaka između prvog i trećeg, a treći je *Padding*, widget za koji je predviđen unos vrijednosti padding. Unutar trećeg widgeta su dvije slike unutar widgeta *Stack* (elementi se slažu jedan na drugog), prva slika *cadrant.png* je podloga kompasa na kojoj su raspoređeni stupnjevi u krug, a druga slika *compass.png* je igla od kompasa. Ona je stavljena unutar widgeta *Transform.rotate* koja rotira svoj *child* widget oko njegove sredine prema određenoj *angle* vrijednosti.

I na kraju ostaje jedan widget i jedna funkcija koje inače ostaju u *main.dart* datoteci dok se ostale raspoređuju u neke druge foldere:

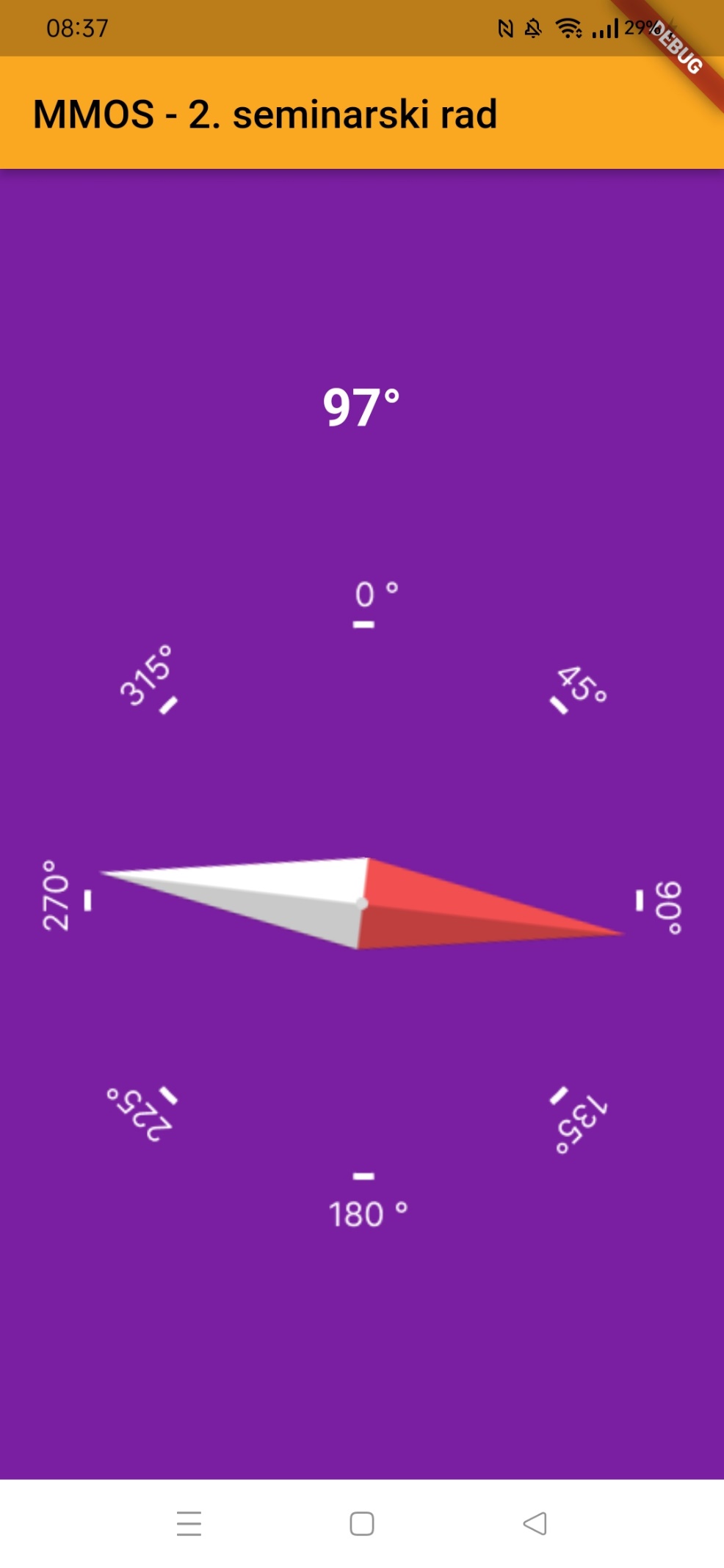
Slika na kojoj se prikazuje tekst

Opis je automatski generiran

Widget *MyApp* je početni widget svake aplikacije, tu se može uređivati stil cijele aplikacije, mogu se postavljati stilovi za tekst, dodavati *provideri* – rješenja za upravljanje stanjem aplikacije itd. *MyApp* vraća widget *MaterialApp* koji je specifičan za Android uređaje, tako umjesto *MaterialApp-a* može biti i *CupertinoApp* koji je specifičan za iOS operacijski sustav. Platforma na kojoj se aplikacija izvodi se može lako provjeriti korištenje Platform klase koju pruža dart.

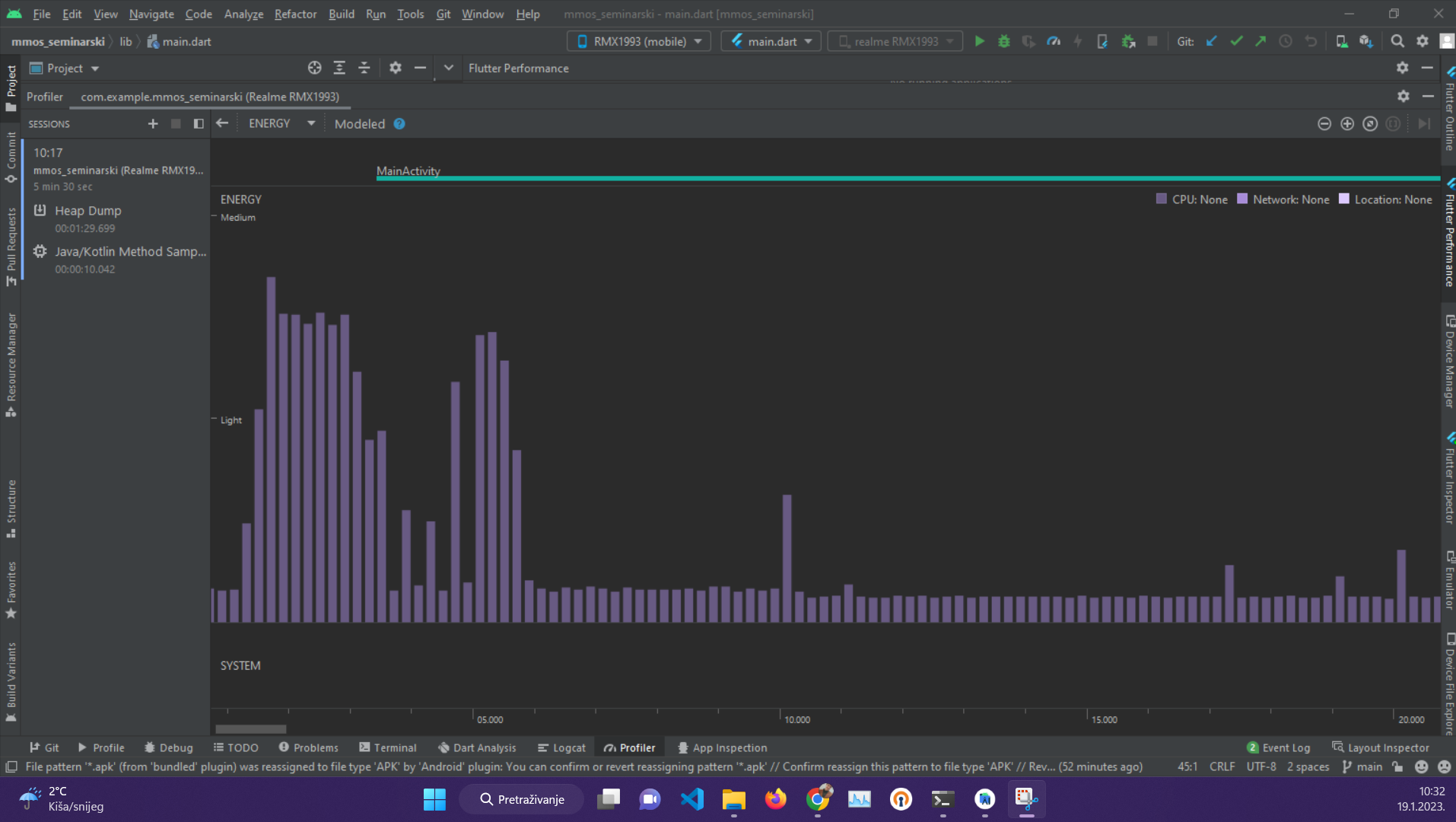
Funkcija main, početna točka aplikacije, u sebi ima metodu runApp koja uzima widget koji joj se proslijedi i stavlja ga na ekran.



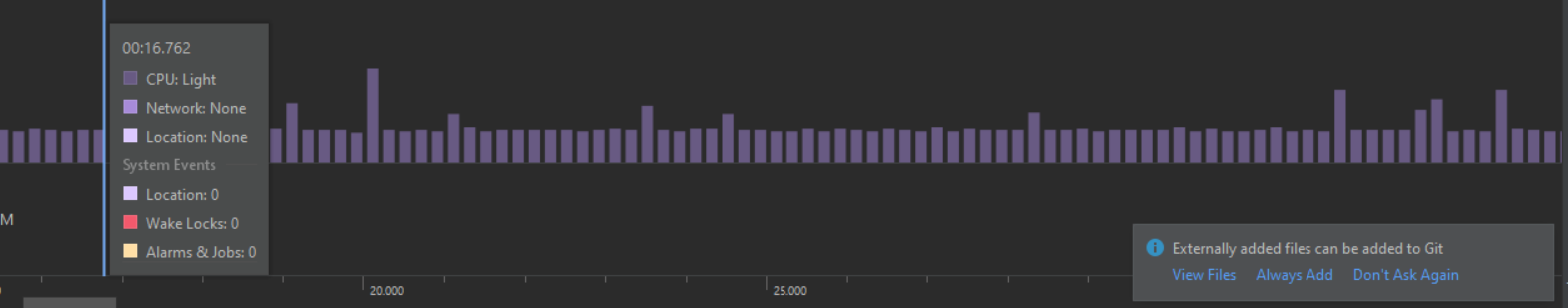


**Profiliranje aplikacije**

Energija



Pri samom pokretanje aplikacije uočava se malo veća potrošnja energije u odnosu na ostatak izvođenja aplikacije gdje je potrošnja minimalna.



Slika na kojoj se prikazuje tekst, monitor, snimka zaslona, na zatvorenom

Opis je automatski generiranCPU

Nakon snimanja potrošnje CPU-a mogu se vidjeti svi događaji koji su koristili procesor kao što su dispatchSensorEvent, onSensorChanged itd. CPU usage je cijelo vrijeme tijekom snimanja varirao između 0 i 1%.

U Bottom Up prikazu možemo vidjeti odakle dolazi koji event, pa tako kod dispatchSensorEventa možemo vidjeti da smo uspješno aktivirali senzor od uređaja

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

Memorija

Prilikom snimanja memorije natrag dobijemo popis svih objekata koji su alocirani na heap-u (gomili) te povratne informacije o heap-u uključujući i podatke o curenju memorije

Slika na kojoj se prikazuje tekst

Opis je automatski generiran