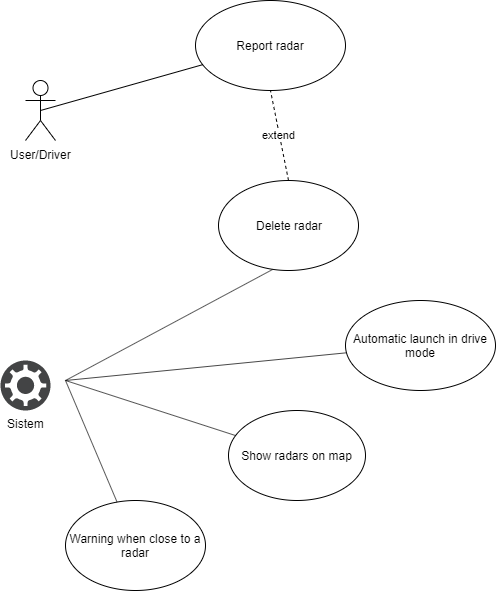
Dokumentacija: ShooDar

# Diagram primerov uporabe



# Shema podatkovne baze

# 

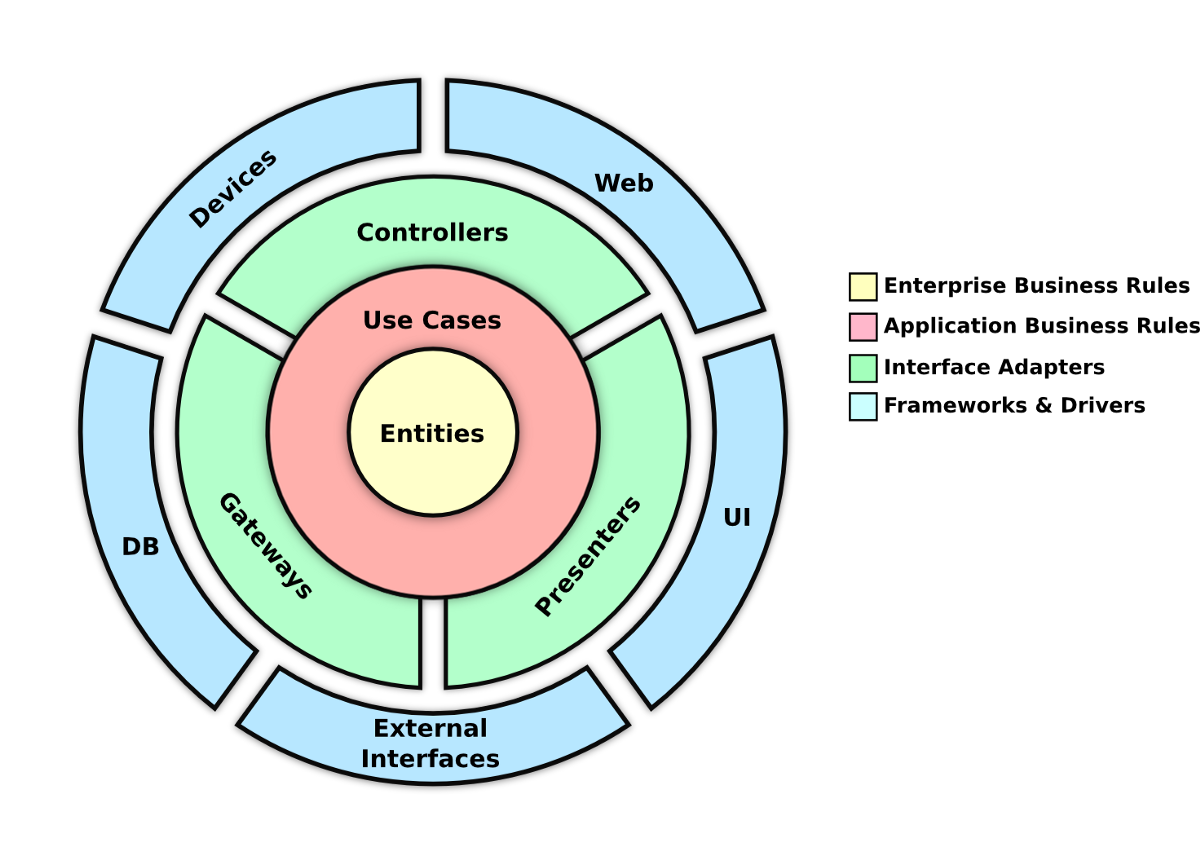
Zaradi enostavne narave naše aplikacije, ki na kompleksne načine predstavi svoje funkcionalnosti vidimo, da je v osnovi podatkovna baza zelo enostavna. Gre za povezavo med uporabnikom in radarjem, saj bo to v bodoče pomembno za indeksacijo in določanje kredibilnosti uporabnikovih dejanj. Tako se bomo rešili nezanesljivih vnosov in povečali zaupanje v naše podatke. Trenutno kot je, mora biti uporabnik prijavljen v sistem, da radar sploh sme dodati.

# Zaslonski posnetki

|  |  |
| --- | --- |
| Zaslonski posnetek | Opis |
|  | Naš osnovni poglej je na zemljevid. Na ta pogled se aplikacija tudi odpre in je naša osrednja funkcionalnost. Tu uporabnik prejema obvestila o lokaciji radarjev in jih lahko tudi sam doda s pritiskom na priročno postavljen gumb z ikono fotoaparata. Obvestila se lahko prejmejo tudi v obliki potisnih obvestil. Ko pride uporabnik zelo blizu, da se lahko prepriča ali je radar še na mestu se ponovno prikaže dialog, ki uporabnika vpraša ali je radar še vedno tam. |
|  | Osnoven obrazec registracije in prijave. |
|  | Napredni pogled na delovanje aplikacije, za katerega mora biti uporabnik prijavljen v sistem. Tu se uporabnik lahko prijavi, ogleda profil (še ni pripravljena funkcionalnost), briše radarje in si ogleduje ali je radar še aktiven na podlagi rdeče/zelene lučke pred radarjem. Namenoma naslednji pogled na podrobnosti radarja ni nakazan, saj služi kot nekakšna manj pomembna funkcionalnost, ki jo naj uporabniki odkrijejo samo in doda nekakšno nagrado za uporabljanje aplikacije dolgoročno. |
|  | Ogled podrobnosti, tu je na zemljevidu prikazana lokacija radarja z nekaj osnovnimi podatki. V planu je dodati še nekaj zanimivih statistik, da bodo uporabniki bolj z veseljem preživljali čas v naši aplikaciji, tudi ko niso na poti. |
|  | Uporabnik ima možnost spreminjati svoje preference pri uporabi aplikacije, s tem lahko močno izboljša življenje baterije ali pa jo popolnoma uniči.  Ker imamo možnost potisnih obvestil je pomembno da ima uporabnik možnost te onemogočiti.  Na voljo je tudi nekaj nastavitev glede enostavnosti upravljanja aplikacije na poti. |

# Dodatno poglavje: Clean Architecture

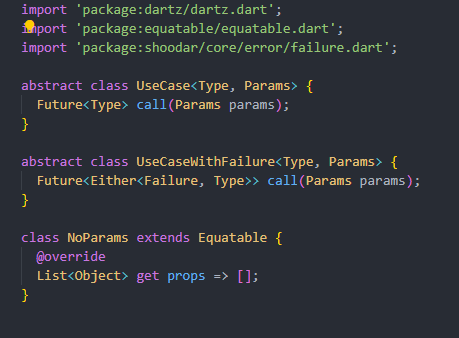
Osnovna ideja: inverzija odvisnosti. Potrebno je ločiti notranjo logiko aplikacije od implementacije. S slike 1 je razvidno, da obstajajo 4je sloji oz. 2 sloja in jedro. V jedru predpišemo primere uporabe v obliki vmesnikov in jih potem implementiramo. Ti vmesniki služijo kot pogodbe, kaj bomo prejeli od nekega razreda, ki implementira ta vmesnik in tako ločimo kodo na logične in obvladljive dele.



Slika : 1, model arhitekture

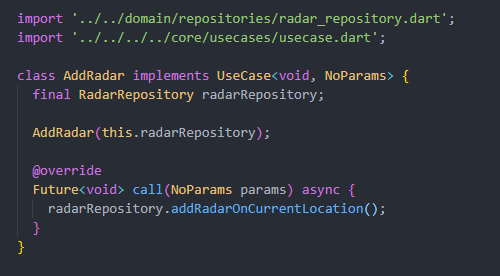
Pa si poglejmo implementacijo naše osnovne funkcionalnosti aplikacije – dodajanje radarjev na zemljevid pri naši trenutni lokaciji.

Prvo je potrebno definirati kako naši primeru uporabe sploh izgledajo:



Slika : 2, definicija oblike primerov uporabe

Nato se to obliko implementira za konkreten primer uporabe:



Slika : 3, konkreten primer uporabe

Za lažje delo s pridobivanjem podatkov, ki so lahko v našem primeru odvisni od povezave s spletom smo implementirali tudi vzorec odlagališča. Ta nam omogoča predpis določene oblike v kateri bomo podatke pridobili. Nato lahko to pogodbo sprejmemo z različnimi razredi. To nam omogoča probleme s povezljivostjo rešiti s pomočjo nekaj pogojnih stavkov in lokalne hrambe podatkov, ko jih imamo enkrat pridobljene in se samo osvežujejo ko je to potrebno.

To je zelo na kratko opisan naš proces dela, saj je pristop čiste arhitekture kar obsežen, vendar koristen. Več informacij je na voljo [tu](https://resocoder.com/flutter-clean-architecture-tdd/).