Turculetu Daniel IV/5.2

Domain Driven Design – Entity

Conceptul DDD este un mod de abordare a sistemelor software complexe, punându-se accentul pe ideea de domeniu sau sferă de activitate în care se implementează o aplicație software. Design-ul unui astfel de sistem se realizează pe baza unui model al domeniului, caracterizat prin proprietăți și comportament și care mai departe duce la definirea unui limbaj universal ce ajută la scrierea codului într-un mod cât mai ușor de înțeles și care să reflecte realitatea domeniului aplicației.

Modelarea DDD se bazează pe următoarele elemente de bază:

o **Entities**

o Value Objects

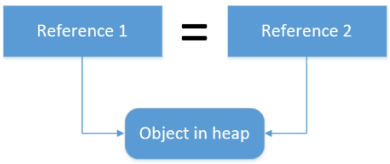
o Aggregations + Roots

Entity – Un obiect ce nu este definit prin atributele sale, ci prin continuitate și prin identitatea sa.

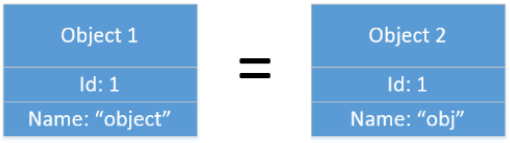
Entitățiile sunt concepte ce pot fii identificate în mod unic. Ele au un aspect ce nu se modifică, ce acționează ca un identificator, dar pe lângă acest aspect imuabil putem modifica tot ce are legătură cu identitatea, dar ea rămâne aceeași.

Entites vs Value Object: tipuri de egalități

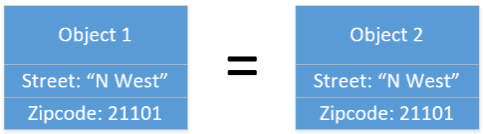
Egalitatea de referințe – două obiecte sunt considerate egale dacă ele trimit către aceeași adresă din memorie



Egalitatea de identificare – se presupune că o clasă are un câmp pentru id. Două instanțe ale acestei clase sunt egale dacă au aceeași identificatori



Egalitatea structurală – două obiecte sunt egale dacă toți membrii acestor obiecte sunt identici



Argumente pro și contra pentru o clasă de entități standard si o clasă cu abordare DDD :

- Avantaje pentru o clasă de entități standard: Simplu. Cod minim

- Avantaje pentru o clasă cu abordare DDD : Foarte observabil, denumiri sugestive. Control bun la accesarea datelor

- Dezavantaje pentru o clasă de entități standard: Un sistem complex devine foarte greu de înțeles. Posibilitate de cod duplicat.

- Dezavantaje pentru o clasă cu abordare DDD: puțin mai mult cod de scris.

Pe langă abordarea DDD exista incă 3 abordări DDD care sunt prezentate in diagrama de mai jos indicand ce cod contine fiecare clasa entitate.

- Repository: un model de depozit oferă un set de metode pentru a accesa baza de date. Aceste metode "ascund" codul necesar implementării diferitelor caracteristici de bază de date.

- Query Objects: Acesta este un model de proiectare pentru construirea de interogări de bază de date eficiente pentru EF.

- C(r)UD: Create, (read), Update and Delete- Folosind termenul C (r) UD pentru a separa citirea de funcțiile care schimbă baza de date (creare, actualizare și ștergere).

- Business logic: procese mai complexe care depășesc validarea simplă. Acestea pot avea calcule complexe (de exemplu, un motor de tarifare) sau pot solicita acces la sisteme externe (de ex. Trimiterea unui e-mail unui client).