Szabó Ágota-Hella
Anul IV, IS

Domain Driven Design – Repository

-Tema 2 PSSC-

Cu evoluarea sistemului informatic și a programelor, a software-elor din ce în ce mai complexe, era nevoie de diferite metode de abordare pentru manipularea cerințelor complexe. Din această cauză s-au creat așa numiți design pattern-uri, care ajută un dezvoltator de software la scrierea unui cod curat, lizibil, ușor de modificat astfel încât la apărarea noilor cerințe dezvoltatorul să se axeze pe rezolvarea problemelor, nu la refactorizarea codului.

În zilele noastre sunt deja o mulțime de design pattern-uri, dintre care se numără și Domain Driven Diesign-ul (scurt: DDD). DDD este o abordare a dezvoltării software pentru nevoi complexe prin conectarea implementării cu un model de evoluție. DDD a fost introdus și popularizat de Eric Evans, în cartea sa apărută în 2004: *Domain Driven Design: Tackling Complexity in the Heart of Software.*

Premisa DDD este următoarea:

- plasarea accentului principal al proiectului asupra domeniului principal și a logicii acestuia;
- bazându-se pe modele complexe pe un model al domeniului;
- inițierea unei colaborări creative între experți tehnici și de domeniu (nu neapărat cel de IT) pentru a rafina iterativ un model conceptual care abordează probleme specifice domeniului.

Câteva termeni esențiali și definiția lor scurtă:

Contextul: Cadrul în care apare un cuvânt sau o afirmație determină semnificația acestuia. Afirmațiile despre un model pot fi înțelese doar într-un context.

Modelul: Un sistem de abstractizări care descrie aspecte selectate ale unui domeniu și poate fi folosit pentru a rezolva probleme legate de acest domeniu

“Ubiquitous Language”: Un limbaj structurat în jurul modelului de domeniu și utilizat de toți membrii echipei pentru a conecta toate activitățile echipei cu software-ul.

Contextul limitat: O descriere a unei limitări (de obicei un subsistem sau a unei echipe specifice) în cadrul căruia un anumit model este definit și aplicabil.

DDD definește, de asemenea, o serie de concepte de nivel înalt care pot fi utilizate împreună pentru a crea și a modifica modele de domenii: *Entity, Value Object, Domain Event, Aggregate, Service, Repositories, Factories.*

Repository

Prin DDD repository se înțelege un serviciu care utilizează o interfață globală pentru a oferi acces la toate entitățile și obiectele de valoare care se află într-o anumită colecție agregată. Metodele trebuie definite astfel încăt să permită crearea, citirea, modificarea și ștergerea obiectelor din agregat (*Aggregate).*

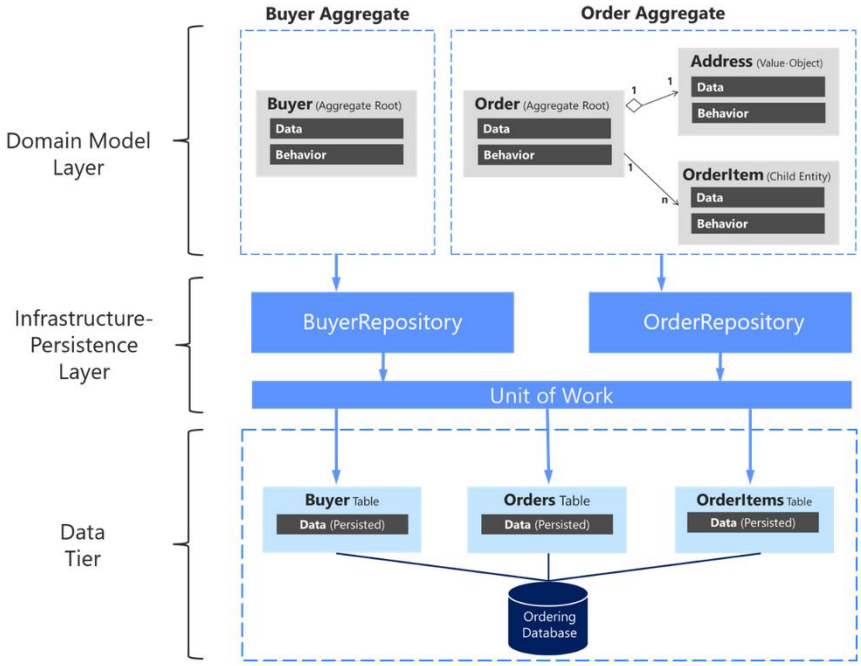
Repository Pattern

Este unul dintre cele mai majore (și cele mai argumentate) modele structurale din DDD. Pe scurt este un set de metode a unei entități care ajuta la comunicarea cu baza de date. De asemenea el permite testarea cu ușurință a aplicației cu teste de unitate.

Un *repository* este o clasă sau o componentă care încapsulează logica necesară pentru accesarea surselor de date. Ele centralizează funcționalitatea comună de acces la date, oferind o mai bună întreținere și decuplarea infrastructurii sau a tehnologiei utilizate pentru a accesa baza de date din stratul de model de domeniu.

Pentru fiecare *aggregate*, trebuie să se creeze o clasă de *repository*. Într-un microserviciu bazat pe DDD, singurul canal care ar trebui utilizat pentru actualizarea bazei de date ar trebui să fie *repository*. Acest lucru se datorează faptului că acestea au o relație unu-la-unu cu rădăcina agregată, care controlează invarianții agregatului și consistența tranzacțională.

Practic, un *repository* permite să se populeze datele din memorie care provin din baza de date sub forma entităților domeniului. Odată ce entitățile sunt în memorie, ele pot fi modificate și apoi trimise înapoi în baza de date prin tranzacții.



*Relația între repositories, aggregates și tabelele din baza de date **

Pe scurt, un repository pattern:

- Nu este un strat de acces la date
- Oferă un nivel mai ridicat de manipulare a datelor
- Este o colecție de rădăcini agregate
- Oferă un mecanism de gestionare a entităților

Bibliografie și linkuri

https://airbrake.io/blog/software-design/domain-driven-design
https://docs.microsoft.com/en-us/dotnet/standard/microservices-architecture/microservice-ddd-cqrs-patterns/infrastructure-persistence-layer-design?view=aspnetcore-2.2#the-repository-pattern
https://lostechies.com/jimmybogard/2009/09/03/ddd-repository-implementation-patterns/

++ https://medium.com/the-coding-matrix/ddd-101-the-5-minute-tour-7a3037cf53b8
https://pehapkari.cz/blog/2018/02/28/domain-driven-design-repository/

*Sursa: https://docs.microsoft.com/en-us/dotnet/standard/microservices-architecture/microservice-ddd-cqrs-patterns/infrastructure-persistence-layer-design?view=aspnetcore-2.2#the-repository-pattern