# Tema 2 PSSC-Repository

**Definitie-Ce este un repository(depozit)?**

**Repository** sau **Repo** – reprezintă locația unde sunt stocate toate fișierele aferente unui anumit proiect. Fiecare software, proiect sau aplicație va avea un „repo" propriu, ce poate fi accesat cu un URL unic.
**Repository,** la fel ca o colecție, are responsabilitatea de a adăuga un obiect, de a obține obiecte în funcție de identificator sau de criterii complexe și, eventual, de a elimina un obiect. Există, de asemenea, cazuri de utilizare care necesită agregări cum ar fi: Câte obiecte sunt în system?, Cantitate totală a tuturor produselor din **depozit**?. Pentru aceste cazuri de utilizare, **depozitul** poate oferi metode de agregare directă, astfel încât nu trebuie să preluăm ineficiența încărcăturilor de obiecte.
**Depozitele** sunt create pentru agregate numai pentru că agregatele reprezintă blocurile noastre, unitățile noastre. De asemenea, lucrează întotdeauna cu întregul agregat, nu numai cu o parte internă, nu cu un agregat parțial, întotdeauna cu întregul agregat.
**Depozitarul** este implementat în stratul de domeniu, deoarece funcționează cu obiecte de domeniu. Dar în stratul de domeniu nu ar trebui să avem nicio idee despre nici o bază de date și niciun spațiu de stocare, deci **depozitul** este doar o interfață.

**Implementare**

Putem stoca obiecte de domeniu într-o bază de date relațională, într-o bază de date a documentelor, într-un sistem extern conectat prin API sau în orice altceva ce ne putem imagina. Toate aceste sisteme sunt infrastructură pentru domeniu, astfel încât implementarea depozitului se află în stratul de infrastructură. Cel mai simplu și mai instructiv este implementarea memoriei. O implementare care păstrează obiecte în timpul unei vieți și nu le persistă deloc.

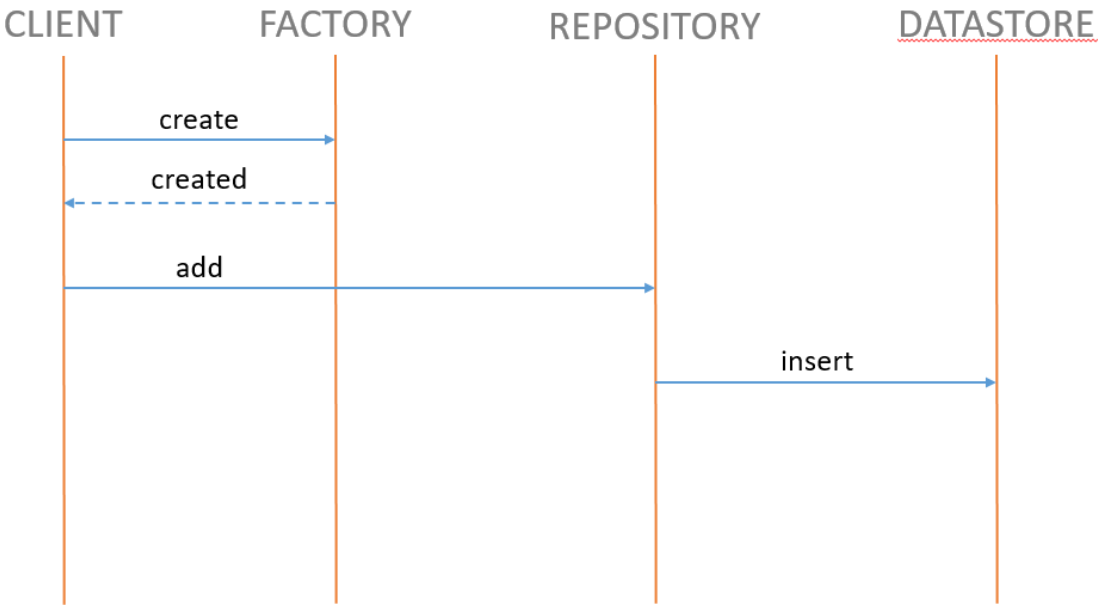
## Teste

Implementarea memoriei este utilă pentru teste complexe sau pentru module. Integrăm mai multe părți ale sistemului, dar nu folosim implementarea reală a depozitului. Deoarece totul este în memorie, testele sunt rapide și încă testează întreaga componentă sau modul.

## Repository

- un singur repository pe aggregation root
- conține cod specific tipului de persistență (CRUD, query)
- singurul loc în care există detalii despre persistență
- elementele de infrastructură se combină cu elementele din domeniu

**Factory vs. Repository**




**Depozite în Servicii de aplicații**

Astfel, într-una din metodele de servicii de aplicație, numim o metodă de serviciu de domeniu (verificarea regulilor de afaceri) și dacă condiția este bună, depozitul este chemat pe o metodă specială pentru a persista / recupera entitatea din baza de date.

```
class ApplicationService{

  constructor(domainService, repository){
    this.domainService = domainService
    this.repository = repository
  }

  postAction(data){

  Entity entity = new Entity(data.name, data.surname);

  this.repository.persist(entity);

  // ...
}
}
```

**Depozite in Servicii de Domenii**

Uneori este util să injectați și să utilizați un depozit într-un serviciu de domeniu, dar numai dacă depozitele  sunt implementate astfel încât să accepte și să returneze numai rădăcini agregate și, de asemenea, în cazul în care abstracționați logica care implică mai multe agregate.

**Exemplu de implementare pentru depozite in servicii de domenii:**

```
doSomeBusinessProcess(name, surname, otherAggregateId) {

  OtherEntity otherEntity = this.otherEntityRepository.get(otherAggregateId);
```

```
  Entity entity = this.entityFactory.create(name, surname);

  int calculationResult = this.someCalculationMethod(entity, otherEntity);

  entity.applyCalculationResultWithBusinessMeaningfulName(calculationResult);

  this.entityRepository.add(entity);
}
```

**Concluzie**

Depozitele sunt colecții persistente și ne permit să presupunem că sistemul este în memorie. Depozitul funcționează cu agregate complete și este o interfață în stratul de domeniu. Implementarea concretă se face prin infrastructura pe care o folosim. Testele pot fi scrise pentru interfața depozitului.