**Tema 2**

**Event Sourcing**

**Event Sourcing** este reprezentat de actiunea pe care o face, adica asigura salvarea modificarilor starii unei aplicatii intr-un obiect eveniment in ordine cronologica. Nu se concentreaza pe starea curenta.

Acesta poate fi folosit atunci cand se proiecteaza logica de persistenta a unei cereri, el fiind un model architectural.

Informatiile de stare se stocheaza in “event store”. Evenimentele se salveaza ca intr-o stiva, doar ca se poate accesa oricand oricare eveniment anterior, deoarece informatiile nu se suprascriu. Care rezultat, toata istoria este mentinuta intacta. Locul in care se salveaza informatiile, Event Store, are proprietate in operatiuni care includ: verificarea, corelarea, detectarea rapida, corectia si reconstituirea.

**Intrebuintari**

Obiectele eveniment initiaza modificarile obiectelor de domeniu, avand urmatoarele facilitati:

* Interogare temporala: este usor de vizualizat starea aplicatiei in orice moment
* Eveniment Replay: in urma detectarii unui eveniment trecut incorrect, se calculeaza consecintele inversarii acestuia si ale evenimentelor de mai tarziu, apoi reia evenimentul nou.
* Rebuil complet: Din jurnalul de evenimente se vor rula evenimentele pe o aplicatie goala

**Avantaje**

1. Modele expresive: modelarea evenimentelor ca obiecte de prima clasa, modelele semanand cu procesele modelate, existant claritate

2.Comunicare cu expertii de domeniu foarte la indemana

3.Evenimentele sunt simple obiecte care descriu unele actiuni care au aparut, impreuna cu toate datele associate, necesare pentru a descrie actiunea reprezentata de eveniment. Acestea pur si simplu sunt inregistrate pentru utilizarea ulterioara. Asadar sunt simplificate implementarea si gestionarea.

4.Stucturi de date efemere: Se pot proiecta datele in forma in care avem nevoie.

5.Event sourcing poate preveni conflictele in urma actualizarilor, deoarece evita cerinta de a actualiza direct obiectele din spatiul de stocare.

6.Evenimente imuabile

7.Baze de date standard

8.Servicii compuse cu usurinta: Adaugand noi functionalitati, nu este necesara modificarea codului existent.

**Dezavantaje**

## Actualizarea evenimentelor: se vor schimba de-a lungul timpului, iar aste lucru poate fi dificil de manevrat daca nu vom planui acest lucru din timp.

## Developerii au nevoie de reprogramare. Se manifesta eventimente care sunt CRUD si nu reflecta limbajul domeniului.

## Eventuala consistenta: Datele din proiect nu sunt imediat actualizate.

**Aplicabilitate**

* Cand este necesar sa se evite partial sau complet aparitia unor conflicte dupa actualizari
* La dorinta de a captura intentia, scopul sau motivele in date, spre exemplu: modificarile aduse clientului, datele pot fi capturate ca o serie de evenimente specific, cum ar fi mutatul intr-o alta casa, inciderea unui cont bancar, un deces, etc.
* Se gasesc la sisteme multi-cerere pe mai multe domenii si regiuni, partajarea acelorasi reguli, conventii si politici despre evenimentul de manipulare.

Exemplu cont bancar

|  |  |  |
| --- | --- | --- |
| Comanda | Eveniment | Suma curenta |
| CreareCont | ContCreat | 0 |
| Depunere 100lei | Depus 100lei | 100lei |
| Depunere 250lei | Depus 250lei | 350lei |
| Retragere 150lei | Retras 150lei | 200lei |
| Retragere 250lei | Suma depasita | 200lei |

Observam ca starea curenta a unui obiect se schimba, iar secventa de evenimente care a cauzat obiectul este in starea curenta. Modificarile se stocheaza in coloana a doua. Evenimentele se reiau pana se ajunge la starea actual din nou.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | CreareCont | | Depunere aprobata | | Retragere aprobata | | Retragere respinsa |   Event store    Evenim |  | IncarcaEvenim()  EvenimInregistrate()   |  | | --- | | CreareCont | | Depunere aprobata | | Retragere aprobata | | Retragere respinsa | |