UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI

**FACULTATEA DE INFORMATICĂ**



LUCRARE DE LICENȚĂ

**Avantajele patternului Event Sourcing**

**propusă de**

***Gabriel-Angelo Pantiru***

**Sesiunea:** *iulie, 2018*

**Coordonator Stiintific**

**Colab. Florin Olariu**

**UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IASI**

**FACULTATEA DE INFORMATICA**

**Avantajele patternului Event Sourcing**

***Gabriel-Angelo Pantiru***

**Sesiunea:** *iulie, 2018*

**Coordonator științific**

***Colab. Florin Olariu***

DECLARAŢIE PRIVIND ORIGINALITATE ŞI RESPECTAREA DREPTURILOR DE AUTOR

Prin prezenta declar că Lucrarea de licenţă cu titlul „*Avantajele patternului Event Sourcing*” este scrisă de mine şi nu a mai fost prezentată niciodată la o altă facultate sau instituţie de învăţământ superior din ţară sau din străinătate. De asemenea, declar că toate sursele utilizate, inclusiv cele preluate de pe Internet, sunt indicate în lucrare, cu respectarea regulilor de evitare a plagiatului:

* toate fragmentele de text reproduse exact, chiar şi în traducere proprie din altă limbă, sunt scrise între ghilimele şi deţin referinţa precisă a sursei;
* reformularea în cuvinte proprii a textelor scrise de către alţi autori deţine referinţa precisă;
* codul sursă, imaginile etc. preluate din proiecte *open*-*source* sau alte surse sunt utilizate cu respectarea drepturilor de autor şi deţin referinţe precise;
* rezumarea ideilor altor autori precizează referinţa precisă la textul original.

Iaşi,

Absolvent *Gabriel-Angelo Pantiru*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(semnătura în original)

DECLARAŢIE DE CONSIMŢĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul „*Avantajele patternului Event Sourcing*”, codul sursă al programelor şi celelalte conţinuturi (grafice, multimedia, date detest etc.) care însoţesc această lucrare să fie utilizate în cadrul Facultăţii de Informatică.

De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea „Alexandru Ioan Cuza” din Iași, să utilizeze, modifice, reproducă şi să distribuie în scopuri necomerciale programele-calculator, format executabil şi sursă, realizate de mine în cadrul prezentei lucrări de licenţă.

Iaşi,

Absolvent *Gabriel-Angelo Pantiru*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(semnătura în original)

Introducere

Am ales aceasta tema pentru lucrarea de licenta deoarece Event Sourcing este un pattern care poate oferi foarte multe beneficii. In cele mai multe aplicatii este suficient sa putem interoga baza de data si sa aflam ultima stare a ei. Sunt insa multe cazuri unde ar fi avantajos sa salvam toate starile prin care a trecut aplicatia deoarece acest lucru poate oferi posibilitati de analiza a modului cum se schimba starea aplicatiei ceea ce ar putea aduce beneficii aplicatiilor din anumite domenii cum ar fi cel economic. Pentru a ilustra aceste avantaje am ales sa implementez o aplicatie folosind acest pattern oferind in acest fel explicatii sa le leg de scenarii reale de folosire.

“Event Sourcing este o modalitate de a persista starea aplicatiei tale folosind, stocand o istorie din care se poate determina starea curenta a aplicatiei“[1] .

Patternul de event sourcing este folosit frecvent impreuna cu CQRS si acesta poate fi benefic deoarece permite separarea modelului de citire si modelului de scriere, oferind in acest mod posibilitatea de a lucra la eficienta operatiilor de citire si a operatiilor de scriere in mod separat, lucru care este util in aplicatiile in care se fac foarte multe operatii de citire dar putine de scriere sau invers si este necesara eficientizarea doar a unui subset de operatii care sunt mai importate pentru business.

In event sourcing un agregat reprezinta entitatea de care sunt legate un set de evenimente si fiecare eveniment legat de acel agregat reprezinta o versiune a entitatii.

Unul alt beneficiu al acestei arhitecturi este ca aceste evenimente pot fi folosite pentru restaurarea starii aplicatiei la un anumit moment din trecut.

De asemenea evenimentele pot fi procesate oricand de consumere si folosite in scopuri precum: reconstituirea ultimei stari a aplicatiei, analiza modului in care starile aplicatiei se schimba, determinarea celor mai folosite functionalitati (cu cat sunt mai folosite cu atat prezinta un interes mai ridicat pentru stakeholders).

Implementarea aplicatiei

In implementarea solutiei aplicatiei am folosit de asemenea patternul de CRQS si o librarie ajutatoare pentru aceasta arhitectura care furnizeaza o serie de interfete si functionalitati comune.

Flow-ul acestei implementari va fi: cand un ultilizator modifica un agregat acesta v-a trimite la server modificarea. Serverul va crea o comanda care va fi preluata de un command handler si acesta va crea si insera evenimentul in baza de date. Pentru Query voi implementa 2 modalitati de asemenea. Una va fi cand un utilizator face un reqest pentru un agregat, serverul va trece prin toate evenimentele si il va reconstrui in memorie si il va intoarce clientului. Aceasta optiune este mai costisitoare la query, dar mai putin costisitoare la executia comenzilor. Al doilea mod de implementare va fi ca voi rula un serviciu windows care se va uita in permanenta in baza de date si va verifica daca sunt evenimente neprocesate in baza de date. Acest serviciu va contrui separat intro niste tabele noi agregatii si ii va modifica pe masura ce se proceseaza evenimentele. Rezultatul fiind o structura clasica a bazei de date in care in tabele se gaseste doar ultima versiune a acelei entitati. Acest mod de implementare al query-ului este mai eficient pentru query deoarece nu trebuie sa se reconstituiasca in memorie de fiecare daca ultima versiune a agregatului.

Bibliografie

1. https://msdn.microsoft.com/en-us/library/jj591559.aspx