Gruppens medlemmar

Thomas Krut

Inledning

CQRS (Command and Query Resposibility Segregation) med Event Sourcing som designmönster för microservices-arkitektur

Marknadsanalys

-Hur ser marknadens användande ut med den valda tekniken?

Enligt JetBrains *State of Developer Ecosystem*[1] rapporterar 34% av tillfrågade utvecklare att de jobbar med microservices. Av dessa uppger 12% att de använder CQRS som designmönster. Detta är en ökning med 3% under de senaste tre åren enligt samma undersökning. Vid en snabb genomsökning av tjugotalet jobbannonser inom utveckling på LinkedIn nämns microservices i runt hälften, och CQRS i två.

- Hur används denna teknik i dagens marknad?

CQRS är en teknik för att separera läsning (*Query)* och skrivning (*Command)* av data i applikationer med komplexa datamodeller. Man modellerar då domänen på olika sätt för läsning och skrivning, till skillnad från en enkel CRUD-modell där alla operationer sker mot en gemensam modell. CQRS är lämpligt att använda när en gemensam datamodell blir svår att hantera och förstå, och även i fall där services för läsning och skrivning behöva kunna skala oberoende av varandra. Tekniken används ofta i eventdriven arkitektur, där skrivning/uppdaterad data ger upphov till events som i sin tur uppdaterar representationerna (*read models*) av datan. Ofta används även en *event store* istället för en traditionell databas, vilket innebär att data sparas som en log av händelser och där man får current state genom att ”spela upp” alla events.

- Vilka är de största företagen eller organisationerna som använder denna teknik?

CQRS används ofta för mindre delar av system och inte för övergripande arkitektur. Det är definitivt inte ett mönster som går att tillämpa på alla problemområden; tvärtom bör det noga övervägas och bara implementeras där det inte introducerar onödig komplexitet. Jag har inte lyckats hitta några specifika exempel på stora företag eller organisationer som använder just CQRS men då microserviceslösningar är såpass vanliga förekommer det rimligtvis med en viss frekvens. Enligt Greg Young, mannen bakom begreppen CQRS, är tekniken vanligt förekommande inom finans och gambling[2]. Även event sourcing används ofta i sammanhang där pengar och kassaflöden hanteras, då man enkelt kan spåra orsaken till ändringar (*audit trail)*

1 https://www.jetbrains.com/lp/devecosystem-2023/

2 https://www.youtube.com/watch?v=Q97BWNCkBt8

- Finns det några framtida trender eller förutsägelser relaterade till denna teknik?

Microservices har varit på uppgång ett tag men också kommit att ifrågasättas som universallösning. En del större aktörer som Amazon Prime fått stor uppmärksamhet när de gått tillbaka till en motolitlösning och sparat miljardbelopp på sänkta driftskostnader[3]. Kurvan för just trenden har nog planat ut. Dock är det ett faktum att microtjänster löser många problem med skalbarhet, tillgänglighet och flexibilitet och att den i många fall utgör rätt val av teknik. Val av designmönster är nästa steg och även där måste det handla om ett noga övervägt beslut. CQRS kommer behövas så länge det finns komplexa datamodeller som kan vinna på en uppdelning mellan läsning och skrivning.

Tidsplan

TASKS	TIDSESTIMERING
VECKA 52	
Läs på om design patterns inom Microservices	8h
Lyssna på podcasts med Greg Young Läs artiklar av Martin Fowler Få koll på begrepp som DDD, event store, event sourcing, saga pattern, CQRS	8h
Läs på om CQRS, gör marknadsanalys, skriv resterande tidsplan	8h
VECKA 01	
Nyårsdagen	
Läs på om domain driven design och ’bounded context’. Läs på om relevanta use cases för CQRS.	4h 4h
Hitta relevant domän/datamodell att implementera mönstret på. Gör ett enkelt ERD-diagram eller liknande. Bestäm use case för applikationen. Skriv en kravspec.	5h 1h 1h 2h
Leta upp olika exempel på implementationer av CQRS med/utan event sourcing. Försök hitta en modell att utgå från, eller kombinera flera. Rita upp ett schema över microservices och relevanta events för applikationen.	6h 2h
Välj message broker (RabbitMQ, Kafka, Nats). Ej huvudfokus i projektet så bör vara lätt att sätta upp och komma in i. Välj databas för event store. SQL eller dedikerad databas (EventstoreDB)? Titta på klientbibliotek för broker och databas och välj språk att skriva microservices i.	3h 2h 3h

3 https://www.infoq.com/news/2023/05/prime-ec2-ecs-saves-costs/

VECKA 02	
Implementation av mikroservices för skrivning till event store Docker compose för service och databas + volume.	7h 1h
API endpoints för skrivning av data. Setup av message broker för att hantera events vid skrivning av data.	4h 4h
Läs på om read models och projections. Eventuellt val av en eller flera databaser för detta. Börja implementera projector/read models	4h 4h
Fortsätt implementera projector/read models. API endpoint för queries/läsning av data	4h 4h
Enhetstester i den mån de saknas. Manuella tester med postman för att säkerställa funktionalitet av befintliga komponenter.	6h 2h
VECKA 03	
I mån av tid, enkelt frontend i Vue.js för att create/update samt representation av read model	8h
I mån av tid, deployment till något moln för demo vid redovisning	8h
Sammanställ slutrapport	8h
Inlämning av slutrapport	
Redovisning	