ONDERZOEKSVERSLAG EVENT SOURCING

Onderzoek naar Event Sourcing in de context van een CRM Applicatie

Auteurs: OOSE-team Thompson, studenten HBO-ICT Opleiding HAN

Opdrachtgever: JDI

Inhoudsopgave

Inhoud

Abstract

Dit onderzoeksrapport is opgesteld door Team Thompson in opdracht van JDI. Doel van het rapport is het onderzoeken van de technologie Event Sourcing met daarbij de CQRS architectuur er het concept Event Storage. Voor dit onderzoeksrapport is de volgende hoofdvraag opgesteld: "Hoe kan Event Sourcing i.c.m. CQRS een goede toevoeging zijn op de projecten van JDI?".

Om antwoord te kunnen geven op de hoofdvraag, worden de volgende deelvragen onderzocht:

1. Wat is Event Sourcing?
2. Wat is CQRS?
3. Wat zijn Event Stores?
4. Wat is het voordeel van Event Sourcing?
5. Wat is het nadeel van Event Sourcing?
6. Wanneer is het handig om Event Sourcing toe te passen?
7. Hoe pas je Event Sourcing toe in een applicatie?
8. Op welke projecten van JDI zou Event Sourcing goed kunnen worden toegepast?

Voor dit onderzoek is er gebruik gemaakt van literatuur- en praktijkonderzoek. Eerst zijn de verschillende technieken theoretisch onderzocht en daarna zijn deze toegepast in een prototype om ze te testen. Uit het onderzoek is gebleken dat het gebruik van Event Sourcing en de bijbehorende technieken erg handig kan zijn. Je kunt door middel van Event Sourcing gemakkelijk de historie van een object bijhouden en zo eenvoudig 'terug in de tijd' gaan wanneer er bijvoorbeeld fouten bij een object optreden. Ook het gebruik van CQRS is handig. Hiermee splits je de lees- en schrijfkant van je applicatie op. Je kunt er dan ook voor kiezen om bijvoorbeeld gebruik te maken van twee databases: één voor leesbewerkingen en één voor schrijfbewerkingen. Op het gebied van Event Storage kun je ook alle kanten op. Je kunt bijvoorbeeld gebruik maken van een relationele database of van speciaal voor Event Sourcing ontwikkelde databases, zoals EventStoreDB.

Inleiding

Dit onderzoeksrapport is voortgekomen uit de wens van JDI om onderzocht te hebben wat Event Sourcing is en of dit een goede toevoeging is aan hun vakkennis. Er is onderzoek gedaan naar Event Sourcing en deze techniek is getest in een voorbeeld CRM-applicatie. Doel van het onderzoeksrapport is om een onderbouwd advies te geven aan JDI betreffende het wel of niet toepassen van Event Sourcing in hun projecten.

Het doel van dit onderzoeksrapport is bereikt wanneer er antwoord is gegeven op de volgende hoofdvraag:

Hoe kan Event Sourcing i.c.m. CQRS een goede toevoeging zijn op de projecten van JDI?

Om antwoord te geven op de hoofdvraag, wordt er onderzoek gedaan naar de volgende deelvragen:

1. Wat is Event Sourcing?
2. Wat is CQRS?
3. Wat zijn Event Stores?
4. Wat is het voordeel van Event Sourcing?
5. Wat is het nadeel van Event Sourcing?
6. Wanneer is het handig om Event Sourcing toe te passen?
7. Hoe pas je Event Sourcing toe in een applicatie?
8. Op welke projecten van JDI zou Event Sourcing goed kunnen worden toegepast?

1. Materialen en methoden

In dit hoofdstuk worden de verschillende onderzoekstechnieken toegelicht die zijn gebruikt om antwoord te geven op de hoofdvraag en de deelvragen.

1.1 Literatuuronderzoek

Voor het onderzoek wordt gebruik gemaakt van literatuuronderzoek. Hierbij worden er online, relevante bronnen opgezocht die te maken hebben met hetgeen waar de bijbehorende deelvraag over gaat. Alle bronnen worden gecontroleerd op de volgende punten en worden per bron toegelicht in het hoofdstuk 'Discussie'.

- Actualiteit
- Auteur
- Meningen
- Onpartijdigheid van de auteur

Hieronder de meetbare criteria waaraan de bovenstaande punten moeten voldoen.

Actualiteit

Een bron mag niet ouder zijn dan 2 jaar op het moment van raadplegen *of* de informatie in de bron moet nog steeds relevant zijn, d.w.z. dat hetgeen wat in de bron staat in de loop van tijd niet veranderd kan zijn. Dit laatste moet verantwoord (bewezen) worden in het hoofdstuk 'Discussie'.

Auteur

De auteur moet voldoende kennis hebben over hetgeen waar hij/zij over schrijft. Dat wil zeggen dat de auteur gediplomeerd is in het vakgebied waar de bron over gaat *of* de auteur is minimaal 2 jaar werkzaam in het vakgebied waar de bron over gaat. De kennis van de auteur moet worden verantwoord in het hoofdstuk 'Discussie'.

Meningen

Alle bronnen moeten gecontroleerd worden op meningen. Bronnen moeten objectief zijn. Bronnen mogen alleen een vorm van mening bevatten wanneer er wordt gesproken over voor- en nadelen, omdat dit altijd een persoonlijke mening is. Echter moeten er bij dit soort bronnen minimaal 2 andere bronnen hebben, die ook voldoen aan alle andere beoordelingscriteria, die dezelfde mening ondersteunen. Gebruik van meningen moet duidelijk (d.w.z. concreet in woorden aan het begin van een alinea) worden vermeld.

Onpartijdigheid van de auteur

De auteur mag geen direct relatie hebben met een partij die belang heeft bij een bron. Een voorbeeld: een bron over de manier van werken bij Tesla mag niet geschreven zijn door de teamleider van Tesla, omdat hij/zij een directe relatie heeft met de inhoud van de bron, waardoor al snel een subjectieve en/of verheerlijkende bron ontstaat. De relatie van de auteur (of de organisatie waar de auteur voor werkt) met de inhoud van de bron moet worden onderzocht en verantwoord worden in het hoofdstuk 'Discussie'.

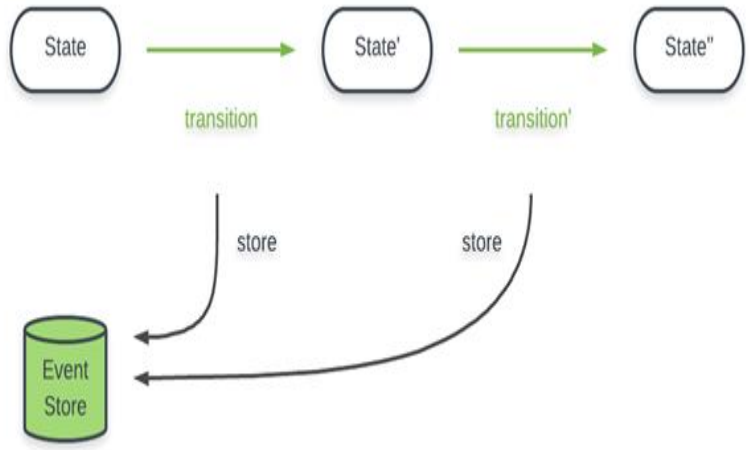
1.2 Praktijkonderzoek

Voor het onderzoek wordt ook gebruik gemaakt van praktijkonderzoek. De techniek achter Event Sourcing zal allereerst worden toegepast in een prototype applicatie. Hierin worden twee varianten van Event Souring toegepast. Nadat de techniek in de praktijk duidelijk is en het verschil tussen de verschillende uitwerkingen van Event Sourcing vergeleken is, zal de techniek worden toegevoegd aan de voorbeeld CRM-applicatie.

2. Wat is Event Sourcing?

Om het eenvoudig te omschrijven: Event Sourcing is het bijhouden van veranderingen om zo te komen tot de huidige staat. Elke verandering is een gebeurtenis (event) die invloed heeft op de meest recente staat van een object. Voorbeeld: de prijs van een product wordt met €5 verhoogt. De verhoging van €5 is dan het event. Hiermee veranderd de huidige staat van het product (prijs X + €5 = nieuwe huidige prijs).

Door het opslaan van alle events, is het mogelijk om op elk moment terug te keren naar de staat van een object op een gegeven datum en tijd. Zo kun je bijvoorbeeld kijken wat de prijs van een product was op 1 december over de afgelopen 5 jaar of zien hoeveel de prijs van een product is gestegen ten opzichte van 6 maanden geleden.



Afbeelding 1: Het concept van Event Sourcing.

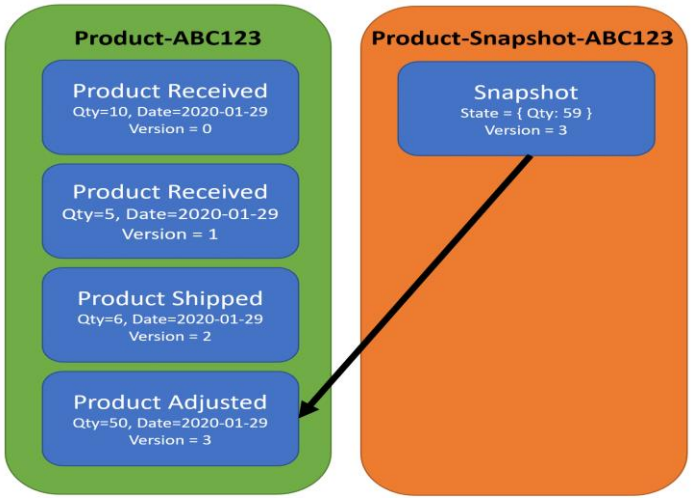
Algemeen geldt dat een reeks aan events altijd waterdicht is. Hiermee wordt bedoeld dat er nooit zomaar een verandering kan zijn geweest die niet is vastgelegd in een event, dit zou namelijk de waarden van de huidige staat onbetrouwbaar maken.

3. Wat is CQRS?

CQRS staat voor *Command and Query Responsibility Segregation* en is een patroon voor het scheiden van lees- en schrijfbewerkingen. Bij CQRS maak je gebruik van zogehete *commands* voor schrijfbewerkingen en *queries* voor leesbewerkingen. De bron (vaak database) waar de informatie naartoe wordt gestuurd / vandaan gehaald wordt, wordt vaak gescheiden bij CQRS. Zo kun je bijvoorbeeld een database hebben voor het opslaan van events en is er een andere database voor het opvragen van data. Deze 'lees database' kan zelfs een andere structuur hebben dan de event database (Microsoft, 2021).

Een voordeel hiervan is dat je de belasting van één database verspreid over 2 databases. Je kunt je voorstellen dat er mogelijk veel meer leesbewerkingen dan schrijfbewerkingen voorkomen in een applicatie. Een ander voordeel is dat de structuur van beide databases afwijkend kan zijn. Je hoeft zo dus geen ingewikkelde queries uit te voeren op de database waarin je informatie uit 5 verschillende tabellen moet opvragen, maar je kunt deze gegevens samen stoppen in de lees database en zo de complexiteit en intensiviteit van je query verlagen. Je verhoogt met het afsplitsen dus de efficiëntie en gelijktijdig behoud je de originele structuur in je 'hoofd database'.

Event Sourcing en CQRS gaan ontzettend goed samen. Dit komt doordat je bij Event Sourcing niet wil dat bij een request alle events opnieuw moeten worden afgespeeld om tot het huidige resultaat te komen. Door gebruik te maken van een aparte lees database (of eventueel aparte tabel), kun je het opgebouwde resultaat apart opslaan. Dit wordt een *Snapshot* genoemd. Door gebruik te maken van deze snapshots, verhoog je de efficiëntie van je applicatie significant. Je kunt bijvoorbeeld de regel stellen dat er na elke 5 events van een object, een nieuwe snapshot wordt gemaakt.



Afbeelding 2: Na een aantal events, wordt er een Snapshot gemaakt.

Stel dat een bepaald object 100 events heeft. Het systeem gaat door alle events heen en bouwt zo toe naar de huidige staat. Deze staat wordt opgeslagen in een snapshot. Hierna doet een gebruiker nog één actie, waardoor er weer een nieuw event wordt toegevoegd.

Als een andere gebruiker nu de data van dit object wil ophalen, hoeft het systeem alleen de snapshot te pakken en hier nog de verandering van het laatste event op toe te passen, om het resultaat te tonen. Dit is veel efficiënter dan als het systeem elke keer door alle 101 events heen moet gaan om tot het resultaat te komen.

4. Wat zijn Event Stores?

Event Stores zijn databases die geoptimaliseerd zijn voor het opslaan van events. Dit kan een SQL database of een NoSQL database zijn. Er zijn NoSQL databases specifiek hiervoor gemaakt, zoals EventStoreDB. De verschillende opties delen dezelfde best-practices, zoals dat de events onaantastbaar zijn, er gebruik wordt gemaakt van snapshots of vergelijkbare technieken en dat het tijdstip van het event en het tijdstip van de toevoeging worden opgeslagen.

Belangrijke principes die gebruikt worden binnen Event Stores en Event Sourcing zijn: Events, Streams, Projections en Subscriptions.

Events

Een event is een gebeurtenis dat plaats heeft gevonden binnen het domein. Events binnen event sourcing bevatten normaal gezien metadata zoals de tijdstip waarop een event plaats vond, een uniek id van het subject en de bijhorende domein data. Deze events worden opgeslagen in een event store database. De events zelf zijn onaantastbaar de uitkomst van de events kunnen aangepast worden door events die op een later tijdstip plaats vinden (Event Store, z.d.).

Streams

Streams binnen event stores zijn een verzameling van opeen volgende events die plaats hebben gevonden op een domein object. Elk event heeft een unieke plek in de stream en kan niet worden aangepast deze plek is aangeduid doormiddel van een oplopend nummer. Een stream zelf heeft ook een uniek id dat het specifieke domain object waar het betrekking toe heeft aanduid.

Events stores zijn er op gemaakt dat er grote aantallen event streams kunnen worden opgeslagen. Er wordt echter wel aan geraden dat het aantal event in een stream niet te hoog wordt de best practice hiervoor is dat streams van korte duur zijn met veel events, of van lange duur met minder events (Event Store, z.d.).

Projections

Projections binnen event sourcing geven een beeld van het onderliggende data-model dat gebaseerd is op de events die hebben plaatsgevonden. Projections binnen Event Sourcing worden ook wel View Models of Query Models genoemd. Vaak bevatten ze de logica om Write models om te zetten naar Read Models.

Projections binnen een Read model wordt gedaan door de evenst die aangemaakt zijn in de Write model om te zetten naar een Read Model view dit object kan dan worden opgeslagen in een andere database of gebruikt worden binnen het systeem om query's uit te voeren.

Binnen de Write Model worden projections (hier ook wel stream aggregations genoemd) gebruikt om de huidige staat van een model vanuit stream events op te bouwen. Het opgebouwde Projection kan gebruikt worden om de huidige staat te valideren van de stream events. Projections moeten worden gezien als tijdelijk en wegwerp baar (Event Store, z.d.).

Subscriptions

Subscriptions in event stores geven de mogelijkheid waarmee andere services kunnen subscriben op event streams. Deze krijgen een notificatie als er een specifieke event binnen een stream wordt opgeslagen in de event store database. Elke event dat wordt opgeslagen in de database kan een notificatie aanmaken. Een belangrijk onderdeel om te onthouden is dat hoewel event stores deze subscribe functie hebben ze vooral geoptimalisseerd zijn voor het opslaan van events. Voor cross-service communicatie kan er beter gebruik worden gemaakt met een gespecialiseerde streaming oplossingen zoals Kafka of Kinesis (Event Store, z.d.).

5. Wat is het voordeel van Event Sourcing?

Dit hoofdstuk bevat meningen. Zie voor meer informatie Hoofdstuk 12: Discussie.

Zoals in hoofdstuk 2 omschreven, is Event Sourcing handig om de historie van veranderingen van een object bij te houden. Deze techniek is uiterst handig om toe te passen op objecten waarin met enige regelmaat de huidige staat kan veranderen en/of waarvan je elke verandering vastgelegd wil hebben. Denk hierbij aan het bijhouden van prijzen en voorraad van een product, of het bijhouden van de tracking informatie van een pakketje.

Door de volledige historie op te slaan, kun je op elk moment terugkijken in de tijd. Stel dat een product ineens een negatieve voorraad van -120 heeft, dan kun je eenvoudig terugkijken in het bij-/afboeksysteem, welke verandering(en) heeft/hebben geleid tot de huidige staat. Ook is het mogelijk om vergelijkingen te maken met waarden uit het verleden.

Events kunnen nooit worden aangepast of verwijderd, het is ten slotte gewoon een tijdlijn van de geschiedenis van een object. Hierdoor is de historie van een product altijd veilig. Mocht een systeem gehackt worden, waarin de voorraad wordt verstoord, dan is het *piece of cake* om het systeem achteraf terug te zetten naar het punt in tijd voordat het systeem gehackt werd *(Sullivan, 2020*).

Het voordeel van Event Souring is dus het altijd bij de hand hebben van de volledige historie van een object, waarmee de huidige staat kan worden nagebouwd. Het is handig om fouten in de huidige staat te vinden en om bedrijfsrapporten te creëren met vergelijkingen van resultaten uit het verleden (*Sullivan, 2017*). In het prototype zit bijvoorbeeld een functie waarmee de status van een bestelling kan worden opgehaald. Zo kun je precies zien waar de bestelling nu is in het proces, maar ook welke stappen deze al heeft ondergaan en wanneer dit was.

6. Wat is het nadeel van Event Sourcing?

Dit hoofdstuk bevat meningen. Zie voor meer informatie Hoofdstuk 12: Discussie.

Het grote nadeel van Event Sourcing is de omvang van de techniek. Hoewel het bijhouden van enkele events voor een object redelijk eenvoudig klinkt, kun je je voorstellen dat je een enorme set aan data krijgt als je voor 10.000 object instanties de geschiedenis gaat bijhouden, als deze individueel 3x per dag zouden veranderen.

Naast de enorme set aan data, wordt het na verloop van tijd ook lastiger om de structuur van een event aan te passen. Wil je extra data opslaan in een event, dan blijf je met lege waarden over in je al bestaande events. Het oplossen van dit probleem kan goed te doen zijn, maar vereist een goede aanpak strategie en heldere event-structuur (*Sullivan, 2017*). Het toevoegen van data aan een event achteraf, was ook een probleem waar tegenaan gelopen werd tijdens de ontwikkeling van het prototype. Hier moest achteraf nog een 'Event Type' bij. Doordat het testdata betrof, kon het in dit geval simpelweg vervangen worden met geüpdatete data waarin dit wel stond. Echter in een productie database is dit natuurlijk niet mogelijk en zou er hier al een probleem ontstaan zijn. Het goed doordenken van de Event structuur voor de ontwikkeling van de applicatie is dus echt cruciaal.

Een ander nadeel voor het eventueel gebruiken van Event Sourcing is de hoge *learningcurve*. Event Sourcing kan in het begin best lastig zijn om correct toe te passen. Het wordt dan ook niet aangeraden om Event Sourcing toe te passen in een (groot) klantenproject, als het ontwikkelteam nog onvoldoende kennis heeft van de techniek. Beter is om de techniek uit te proberen in een kleine, al bestaande functie óf uit te proberen in een testomgeving, om zo het concept van *Trial and Error* toe te passen (*Dudyzc, 2021*).

7. Wanneer is het handig om Event Sourcing toe te passen?

Het is echt af te raden om Event Sourcing overmatig toe te passen in een applicatie, omdat je zo onnodig veel complexiteit creëert. De vraag luidt dan: wanneer is het wél handig om Event Sourcing toe te passen?

In het algemeen geldt: pas Event Sourcing toe op objecten die met enige regelmaat van staat veranderen én waarvan het ook echt bruikbaar is om de historie van op te slaan, zoals bij de voorbeelden in voorgaande hoofdstukken. Door alleen op deze momenten Event Sourcing toe te passen, voorkom je onnodige complexiteit in je systemen voor het managen van de events en bespaar je veel ruimte, doordat je overbodige datahopen achterwege laat.

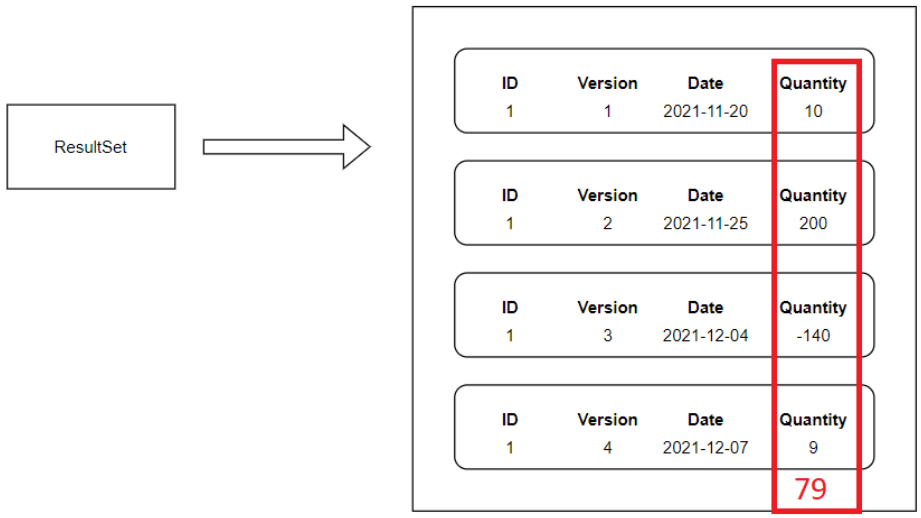
Ook ligt het aan de complexiteit en de omvang van het systeem. Als het gaat om een klein systeem voor het opslaan van wat basisdata voor enkele werknemers, dan is het gebruik van Event Sourcing te complex voor het resultaat wat je ervoor terug krijgt. Gaat het echter om een grote webshop met duizenden klanten, dan is het gebruik van Event Sourcing wel aan te raden.

8. Hoe pas je Event Sourcing toe in een applicatie?

Je past Event Sourcing toe door voor elke gebruikersactie op een object*, de verandering vast te leggen in een gebeurtenis (Event). Deze events worden vervolgens allemaal opgeslagen (in dit geval in een database). Elk event krijgt algemene gegevens mee, vaak zijn dit een versienummer en een datum + tijd waarop het event is aangemaakt. Door middel van het versienummer, kan het systeem de events op de juiste volgorde sorteren.

Vervolgens kan een gebruiker de huidige staat van een object opvragen door óf te kijken naar de waarde bij de meest recente event óf door de events te reconstrueren. Dat wil zeggen dat alle events opnieuw worden afgespeeld en het resultaat daarvan, is dan de huidige staat. Beide varianten zijn getest in het Event Sourcing Prototype.

Het concept van reconstrueren wordt visueel weergegeven in afbeelding 2. Hierin worden de voorraadwijzigingen van een product opgeslagen. Je ziet hier dat het systeem alle events, horende bij een product (ID), opvraagt (ResultSet) en sorteert op het versienummer (Version). Vervolgens kan het systeem deze één voor één langsgaan en de wijzigingen in de voorraad (quantity) optellen en aftrekken om zo tot het uiteindelijke resultaat, de huidige staat, te komen.



Afbeelding 3: Alle events van een product, gesorteerd op versienummer.

** = object waarop Event Sourcing wordt toegepast.*

9. Op welke projecten van JDI zou Event Sourcing goed kunnen worden toegepast?

Om een goed advies uit te brengen over het wel of niet toepassen van Event Sourcing door JDI, wordt in dit hoofdstuk gekeken naar al bestaande projecten van JDI en of Event Sourcing hierop een goede toevoeging zou kunnen zijn. De projecten van JDI zijn hier te vinden. Voor dit hoofdstuk is gekeken naar de cases die online staan op 06-12-2021.

9.1 Case 1: 'Snelheid en comfort voor leden'

In deze staat het ontwikkelen van een CRM applicatie centraal. JDI schrijft hier zelf het volgende over: *"De uitdaging voor deze case lag in het koppelen van de CRM-applicatie middels een API met de Craft website/Intranet"*.

In deze case komen enkele voorbeelden naar voren, waarin het gebruik van Event Sourcing goed van pas zou komen. Om te beginnen, kunnen we al kijken naar het beheer van klanten. Het is ideaal om te kunnen zien of een klant bijvoorbeeld zijn e-mailadres recentelijk heeft veranderd, als je probeert te zoeken op e-mailadres en je de klant in eerste instantie niet kunt vinden.

Een ander goed voorbeeld is het beheren van trainingen. *"Trainingen en webinars voor de leden worden rechtstreek via de API op de website geplaatst. Hier kunnen leden en niet-leden zich geheel geautomatiseerd inschrijven voor een training of webinar. Het CRM-systeem houdt het aantal inschrijvingen bij en communiceert met de website het aantal nog beschikbare plaatsen."*. Ook dit is een ideaal voorbeeld waarop Event Sourcing kan worden toegepast. Zo kun je bijvoorbeeld de historie van een training zien, waarin je een overzicht krijgt met wie zich heeft ingeschreven en daarna weer heeft uitgeschreven. Je kunt een klant dan bijvoorbeeld benaderen als je ziet dat een klant zich vaak nog last-minute uitschrijft.

10. Advies

Kijkend naar het literatuur- en praktijkonderzoek wat verricht is en naar de cases van JDI die bekeken zijn, adviseert Team Thompson JDI om gebruik te maken van Event Sourcing, eventueel met CQRS en het gebruik van een (geoptimaliseerde) Event Storage. Wel wordt er sterk aangeraden om voor iedere case te kijken en per UseCase te beoordelen of Event Sourcing een goede toevoeging zou zijn. Event Sourcing is een goede techniek die in veel applicaties naar voren kan komen, zoals in CRM's of voorraadsystemen. Het bijhouden van de historie van een object kan een klant veel inzicht bieden, zoals bijvoorbeeld in de verkoop van een bepaald product. Het leren van Event Sourcing en het goed onder de knie krijgen hiervan kan lastig zijn, maar er wordt wel aangeraden hier binnen het team van JDI mee aan de slag te gaan.

11. Conclusie

Met behulp van de antwoorden op de deelvragen, kunnen we nu antwoord geven op de hoofdvraag: "Hoe kan Event Sourcing i.c.m. CQRS een goede toevoeging zijn op de projecten van JDI?".

Allereerst weten we nu dat het gebruik van Event Sourcing ontzettend handig kan zijn wanneer je de historie van wijzigingen van een object wilt opslaan. Je kunt zo op ieder moment 'terug in de tijd' gaan, om de staat van een object op dat moment te zien. Event Sourcing kan lastig zijn om toe te passen, zeker wanneer het ontwikkelingsteam hier nog niet veel ervaring mee heeft. Echter wanneer het goed geïmplementeerd wordt, heeft het enorm veel voordelen. Wel moet er altijd op gelet worden dat Event Sourcing alleen wordt toegepast op plekken waar het ook daadwerkelijk van pas komt. Zo wil je vermijden dat je de techniek toepast op objecten waarvan de staat amper veranderd of op plekken waar het helemaal niet relevant is om de historie van een object bij te houden. Je voorkomt zo grote bergen aan nutteloze data en het scheelt een hoop implementatiewerk.

Het gebruiken van CQRS is ook een goede toevoeging op het gebruik van Event Sourcing, omdat je hiermee het schrijven en lezen van je data scheidt. Dit is goed, omdat je de data zo ook op 2 verschillende manieren kan opslaan, waardoor je ingewikkelde en intensieve queries kunt vermijden. Als je er helemaal voor wilt gaan, kun je ook nog gebruik maken van speciaal hiervoor ontwikkelde Event Stores, zoals EventStoreDB, welke speciaal zijn ontwikkeld om de optimalisatie van het gebruik van Event Sourcing te verbeteren.

We kunnen dus concluderen dat de onderzochte techniek (rondom) Event Sourcing een goede techniek is om toe te passen in een applicatie. Kijkend naar de voorgaande projecten van JDI kunnen we stellen dat er ook hier enkele cases zijn waar Event Sourcing perfect zou kunnen worden toegepast en we raden JDI dan ook aan om deze techniek te gebruiken. (Zie verder: Hoofdstuk 10: Advies).

12. Discussie

In dit hoofdstuk komt per gebruikte bron de bronverantwoording aan bod.

12.1 Bronverantwoording

(Sullivan, 2017)	
Auteur	De auteur van deze bron is Barry O. Sullivan. Barry is zelfstandig *Lead Developer and Solutions Architect.* Hij heeft een MSc in Computer Science aan de University of Dublin (zie CV). De auteur geeft zelf aan gespecialiseerd te zijn in DDD en Event Sourcing en zijn werkervaring ondersteund deze bewering.
Actualiteit	De bron is geplaatst op 29-aug-2017. Hoewel dit buiten het afgesproken maximum van 2 jaar valt, is deze bron alsnog gebruikt omdat de inhoud nog altijd actueel is.
Meningen	Deze bron bevat de mening van de auteur. Deze mening wordt echter door meerdere bronnen ondersteund en wordt daarom gezien als algemene mening.
Onpartijdigheid auteur	Alhoewel de auteur een duidelijk positieve mening heeft over het gebruik van de genoemde technieken uit de bron, is hij op geen manier gelinkt aan de ontwikkelaars van de technieken.
(Dudycz, 2021)	

Auteur	Oskar Dudycz is een devloper, technische leidinggevende en (software)architect met meer dan 13 jaar ervaring. Hij geeft regelmatig workshops over Event Sourcing en CQRS en werkt als *Developer Advocate* bij EventStore, wat onder andere inhoudt dat hij de kennis van collega's verbreed op het gebied van Event Sourcing.
Actualiteit	De bron is geplaatst op 23-juni-2021. Dit betekent dat de bron op het moment van schrijven nog geen half jaar oud is en dus ruim binnen de gehanteerde 2 jaar valt.
Meningen	Deze bron bevat de mening van de auteur. Deze mening is echter tot stand gekomen na jarenlange ervaring op het gebied van de beschreven techniek, waardoor de bron voldoet aan de eis.
Onpartijdigheid auteur	Alhoewel de auteur een duidelijk positieve mening heeft over het gebruik van de genoemde technieken uit de bron, is hij op geen manier gelinkt aan de ontwikkelaar(s) van de techniek.
(Booster, 2020)	
Auteur	De auteur van deze bron is Booster (Framwork), een initiatief van de Agile Monkeys. De Agile Monkeys zijn een groep Senior Software Engineers met specialisatie in het toepassen van Event Sourcing in hun softwareapplicaties.
Actualiteit	De bron is geplaatst op 30-okt-2020. Dit betekent dat de bron op het moment van schrijven 1 jaar en anderhalve maand oud is. Hiermee valt de bron nog ruim binnen de gehanteerde 2 jaar.
Meningen	Deze bron bevat de mening van de auteur, omdat ze hun ervaring over het gebruik van de genoemde techniek beschrijven. Doordat Agile Monkeys (heeft ge)werkt voor grote bedrijven als eBay, PayPal en Zara, is de professionaliteit en kennis van de auteur op voldoende niveau.
Onpartijdigheid auteur	Alhoewel de auteur een duidelijk positieve mening heeft over het gebruik van de genoemde technieken uit de bron, is hij op geen manier gelinkt aan de ontwikkelaars van de technieken.
(Microsoft, 2021)	
Auteur	
Actualiteit	
Meningen	
Onpartijdigheid auteur	
(EventStore, z.d.)	

Auteur	De auteur van deze bron is Event Store Ltd, een bedrijf wat eigenaar is van EventStoreDB. Dit is een database speciaal ontwikkeld voor de techniek Event Sourcing.
Actualiteit	De datum van de bron is niet bekend. Echter betreft het de algemene informatiepagina over deze techniek, geschreven door de ontwikkelaars van het product waarover geschreven is. Hierdoor mag aangenomen worden dat de informatie actueel is met de staat en werking van het product.
Meningen	Deze bron bevat lichte meningen over het gebruik van Event Sourcing in het algemeen en niet zozeer over het gebruik van hun eigen product, doordat de gegeven meningen ook in andere bronnen voorkomen worden deze meningen buiten beschouwing van de stijlregel gehouden.
Onpartijdigheid auteur	De auteur van deze bron is de letterlijke ontwikkelaar van het product waarover geschreven wordt. Echter betreft de bron alleen informatie over de techniek en wordt er niet op een overmatige positieve manier (sturend) over gepraat. Hierdoor wordt deze bron buiten de stijlregel van onpartijdigheid gelaten.

Literatuurlijst

1. Sullivan, B. O. (2017, 29 augustus). *Event Sourcing: What it is and why it's awesome*. DEV Community. Geraadpleegd op 8 december 2021, van https://dev.to/barryosull/event-sourcing-what-it-is-and-why-its-awesome
2. Dudycz, O. (2021, 23 juni). *When not to use Event Sourcing? - Event-Driven.io*. EventDriven.Io. Geraadpleegd op 8 december 2021, van https://event-driven.io/en/when_not_to_use_event_sourcing/
3. Booster. (2020, 30 oktober). *Redefining Event Sourcing*. DEV Community. Geraadpleegd op 8 december 2021, van https://dev.to/boostercloud/redefining-event-sourcing-1m52
4. Microsoft. (2021, 9 december). *CQRS-patroon - Azure Architecture Center*. Microsoft Docs. Geraadpleegd op 24 december 2021, van https://docs.microsoft.com/nl-nl/azure/architecture/patterns/cqrs
5. EventStore. (z.d.). *Beginner's Guide to Event Sourcing | Event Store*. Geraadpleegd op 20 december 2021, van https://www.eventstore.com/event-sourcing