

Who Am I?

KEY FACTS

Kim Van Renterghem

Developer since 2008

Senior developer @ zf-scalar



kim-van-renterghem

Traditional systems

CRUD



Whenever we change something, we lose the previous state

```

_id: "3497ea4b-9609-4ef9-b80e-b00215bd5758"
OrganizationId: "813b6fca-19b5-43ae-afdd-bffd035e5811"
Name: "Indus Zone"
Description: null
Type: "IndustrialZone"
▶ Address: Object
▶ Access: Object
UpdatedOn: 2023-12-13T09:08:17.173+00:00

```

```

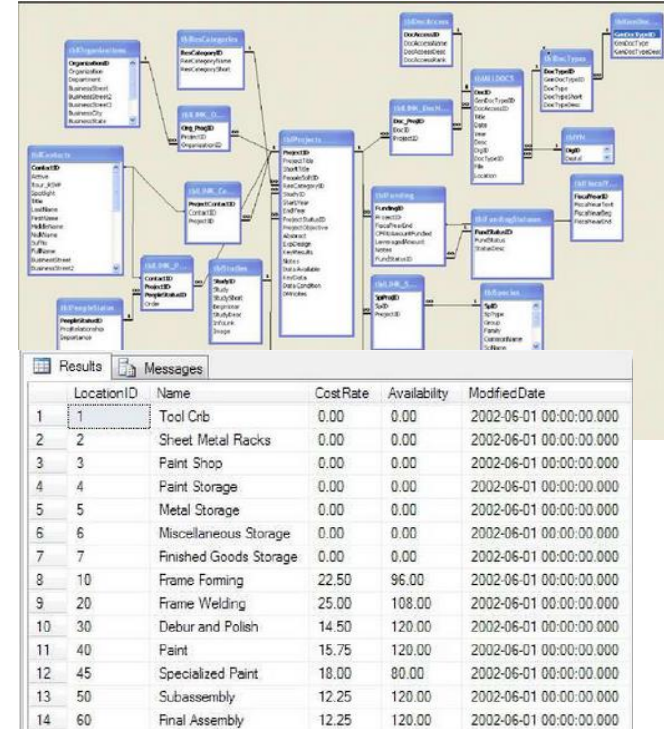
_id: "edcc02f2-37b1-48a8-aa44-0eac01db68e"
OrganizationId: "813b6fca-19b5-43ae-afdd-bffd035e5811"
Name: "chennai"
Description: null
Type: "CustomerSite"
▶ Address: Object
▶ Access: Object
UpdatedOn: 2023-12-13T09:08:17.173+00:00

```

```

_id: "8ffe7b6e-88bb-451a-a637-ca13f5bd4332"
OrganizationId: "813b6fca-19b5-43ae-afdd-bffd035e5811"
Name: "Cust.Site for Aprt"
Description: "updating to send update event"
Type: "CustomerSite"
▶ Address: Object
▶ Access: Object
UpdatedOn: 2023-12-13T09:08:17.173+00:00

```



When you change tracking/detection becomes difficult



Bugs are causing an inconsistent state

=> by correcting you often create a vicious circle



When there are audits, it is impossible to prove that the state is correct.

There are no guarantees that every change is logged because the updates and the audit log do not happen in the same transaction.

Logs

To the Rescue!



Logs

Shows when and what changes

Logs

Logs are difficult to handle because they do not look at the changes but at the consequences of actions.

```

2019-08-22T08:32:32.452Z [INFO] [ ] [factoryHostConfigListener:48] [localhost-startStop-2] - Starting Artifactory [JF_PRODUCT_HOME/Users/yossif/artifactory-oss-7.x-SNAPSHOT]
2019-08-22T08:32:32.490Z [INFO] [ ] [factoryHostConfigListener:283] [art-init] -

```



```

Version: 7.x-SNAPSHOT
Revision: 2147483647
Artifactory Name: /Users/yossif/artifactory-oss-7.x-SNAPSHOT
Node ID: 'yossif-mac'

2019-08-22T08:32:32.544Z [INFO] [ ] [factoryApplicationContext:514] [art-init] - Artifactory application context set to NOT ready by refresh
2019-08-22T08:32:32.548Z [INFO] [ ] [c.m.CertificateMigrationImpl:72] [localhost-startStop-1] - [Certificate Migration] Recreating root certificate to match current certificate version
2019-08-22T08:32:32.553Z [INFO] [ ] [c.m.CertificateMigrationImpl:93] [localhost-startStop-1] - [Certificate Migration] Creating root certificate for version: null
2019-08-22T08:32:32.556Z [INFO] [ ] [org.apache.catalina.startup.HostConfig] [org.apache.catalina.startup.HostConfig] [deployDescriptor] - Deployment of configuration descriptor [/Users/yossif/artifactory-oss-7.x-SNAPSHOT/conf/artifactory.xml]
2019-08-22T08:32:32.899Z [INFO] [ ] [c.m.CertificateMigrationImpl:82] [localhost-startStop-1] - [Certificate Migration] Saved new root certificate in /Users/yossif/artifactory-oss-7.x-SNAPSHOT/conf
2019-08-22T08:32:33.750Z [INFO] [ ] [o.j.s.a.s.r.GrpcServerImpl:59] [localhost-startStop-1] - Starting gRPC Server on port 8045
2019-08-22T08:32:34.489Z [INFO] [ ] [o.j.s.a.s.r.GrpcServerImpl:77] [localhost-startStop-1] - gRPC Server started, listening on 8045
2019-08-22T08:32:34.490Z [INFO] [ ] [AccessServerBootstrapImpl:170] [localhost-startStop-1] - [ACCESS ROOTSTAMP] Updating server 'yossif-mac' private key finger print to: d818c7ed8a2e6bca0b
2019-08-22T08:32:34.211Z [INFO] [ ] [AccessServerBootstrapImpl:300] [localhost-startStop-1] - [ACCESS ROOTSTAMP] No admin user exists - generating an admin user, credentials will be saved to /Users/yossif/artifactory-oss-7.x-SNAPSHOT/conf
2019-08-22T08:32:34.423Z [INFO] [ ] [AccessServerBootstrapImpl:140] [localhost-startStop-1] - [ACCESS ROOTSTAMP] JFrog Access bootstrap finished.
2019-08-22T08:32:36.402Z [INFO] [ ] [c.c.RefreshableScheduledJob:53] [localhost-startStop-1] - Scheduling staleTokenCleanup task to run every 3000 seconds
2019-08-22T08:32:36.404Z [INFO] [ ] [c.c.RefreshableScheduledJob:53] [localhost-startStop-1] - Scheduling loadCertificates task to run every 30 seconds
2019-08-22T08:32:36.404Z [INFO] [ ] [c.c.RefreshableScheduledJob:53] [localhost-startStop-1] - Scheduling FederationCleanupService task to run every 1209600 seconds
2019-08-22T08:32:36.403Z [INFO] [ ] [c.c.RefreshableScheduledJob:53] [localhost-startStop-1] - Scheduling heartbeat task to run every 5 seconds
2019-08-22T08:32:37.300Z [INFO] [ ] [o.j.s.a.s.r.GrpcServerImpl:59] [localhost-startStop-1] - Started AccessApplication in 14.288 seconds (JVM running for 15.854)
2019-08-22T08:32:37.951Z [INFO] [ ] [org.apache.catalina.startup.HostConfig] [org.apache.catalina.startup.HostConfig] [deployDescriptor] - Deployment of configuration descriptor [/Users/yossif/artifactory-oss-7.x-SNAPSHOT/conf/artifactory.xml]
2019-08-22T08:32:37.952Z [INFO] [ ] [org.apache.catalina.startup.HostConfig] [org.apache.catalina.startup.HostConfig] [deployDescriptor] - Deployment of web application directory [/Users/yossif/artifactory-oss-7.x-SNAPSHOT/webapps/artifactory]
2019-08-22T08:32:37.980Z [INFO] [ ] [org.apache.catalina.startup.HostConfig] [org.apache.catalina.startup.HostConfig] [deployDescriptor] - Deployment of web application directory [/Users/yossif/artifactory-oss-7.x-SNAPSHOT/webapps/artifactory]
2019-08-22T08:32:37.981Z [INFO] [ ] [org.apache.coyote.http11.Http11Protocol] [org.apache.coyote.AbstractProtocol] [start] - Starting ProtocolHandler ["http-nio-8045"]
2019-08-22T08:32:37.989Z [INFO] [ ] [org.apache.coyote.http11.Http11Protocol] [org.apache.coyote.AbstractProtocol] [start] - Starting ProtocolHandler ["http-nio-8045"]
2019-08-22T08:32:39.056Z [INFO] [ ] [o.a.s.d.DbServiceImpl:279] [art-init] - Database: Apache Derby 10.14.2.0 - (1328579), Driver: Apache Derby Embedded JDBC Driver 10.14.2.0
2019-08-22T08:32:39.056Z [INFO] [ ] [o.a.s.d.DbServiceImpl:282] [art-init] - Connection URL: jdbc:derby:/Users/yossif/artifactory-oss-7.x-SNAPSHOT/var/data/artifactory/derby
2019-08-22T08:32:39.062Z [INFO] [ ] [o.a.s.d.DbServiceImpl:118] [art-init] - ***Creating database schema***
2019-08-22T08:32:39.740Z [INFO] [ ] [s.r.NodeRegistryServiceImpl:44] [http-mio-8048-exec-1] - join: request to 'join' with serviceId jfrog/mio80b3678d4fj2nyvssq and nodeId yossif-mac
2019-08-22T08:32:39.874Z [INFO] [ ] [o.a.s.d.DbServiceImpl:123] [art-init] - ***Database schema created***
2019-08-22T08:32:40.875Z [INFO] [ ] [s.r.NodeRegistryServiceImpl:44] [http-mio-8048-exec-1] - Scheduling FederationCleanupService task to run every 1209600 seconds
2019-08-22T08:32:40.877Z [INFO] [ ] [f.f.FederationCleanupService:52] [if-access-task] - Hunting clean up outdated Federation events
2019-08-22T08:32:40.256Z [INFO] [ ] [s.r.NodeRegistryServiceImpl:44] [http-mio-8048-exec-1] - join: success returning token with id 222fef3e-4159-475e-357e-d0eb0d49f23
2019-08-22T08:32:40.257Z [INFO] [ ] [s.r.RegistryAuthServiceImpl:80] [http-mio-8048-exec-1] - join request token taken with id 222fef3e-4159-475e-357e-d0eb0d49f23
2019-08-22T08:32:40.330Z [INFO] [ ] [o.j.s.a.s.r.GrpcServerImpl:59] [localhost-startStop-1] - gRPC Server started, listening on 8045
2019-08-22T08:32:40.332Z [INFO] [ ] [RoutingHandler:go:79] [main] - clearing routing dir contents: /Users/yossif/artifactory-oss-7.x-SNAPSHOT/var/data/router/traversal
2019-08-22T08:32:40.332Z [INFO] [ ] [RoutingHandler:go:79] [main] - gRPC server listening on localhost:8045
2019-08-22T08:32:40.332Z [INFO] [ ] [RoutingHandler:go:79] [main] - clearing routing dir contents: /Users/yossif/artifactory-oss-7.x-SNAPSHOT/var/data/router/traversal
2019-08-22T08:32:40.332Z [INFO] [ ] [RoutingHandler:go:79] [main] - clearing routing dir contents: /Users/yossif/artifactory-oss-7.x-SNAPSHOT/var/data/router/traversal
2019-08-22T08:32:40.332Z [INFO] [ ] [RoutingHandler:go:144] [main] - creating router routing file at: /Users/yossif/artifactory-oss-7.x-SNAPSHOT/var/data/router/traversal

```

Logs

They are also not about an entity but about all transactions in a domain; often even about steps we take within a process.

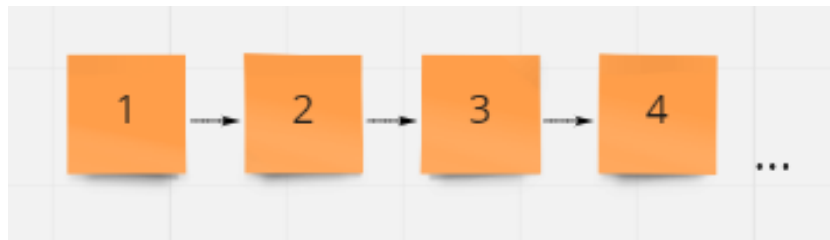
Logs

They show when something changes.
=> But not how or why.

Logs

They do not leave audit trails. These are often added. But in turn are also not reliable.

What if we now used our logs as "the only source of truth"



What if we now used our logs as "the only source of truth"

Let the logs be about the transitions of the past

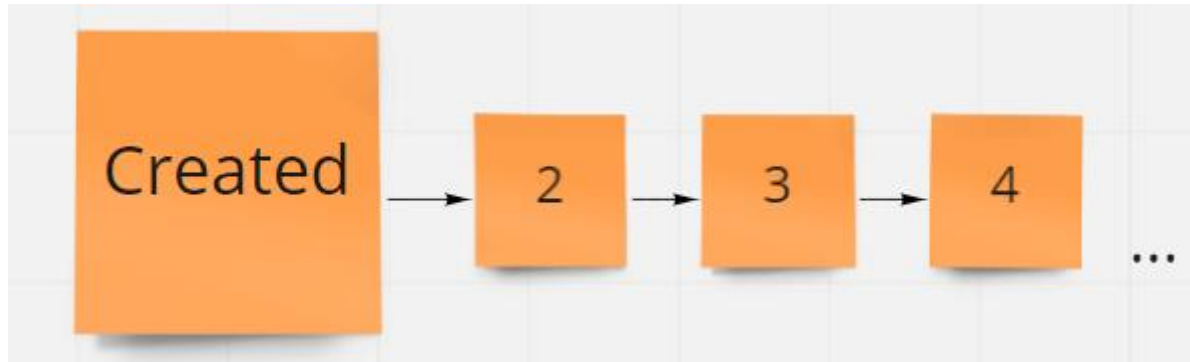
for example:

- Table reserved
- Two martinis ordered
- Martinis delivered
- A steak with fries ordered
- ...
- Bill paid
- Table left

What if we now used our logs as "the only source of truth"

You always start with a created event

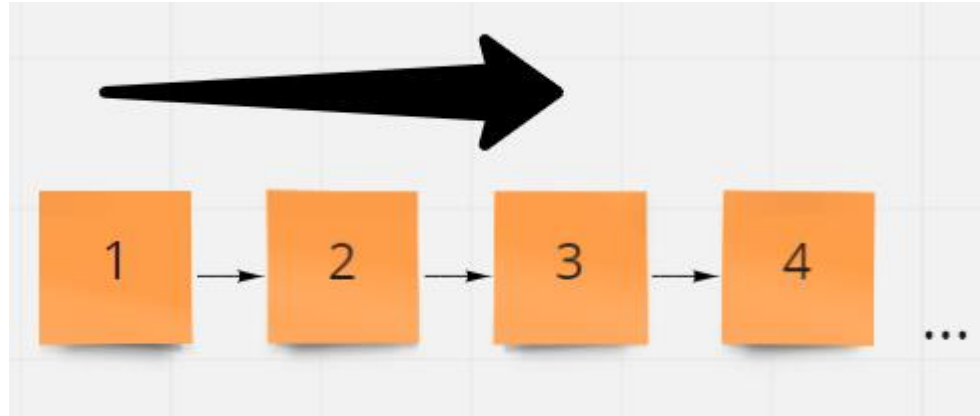
- Table reserved



What if we now used our logs as "the only source of truth"

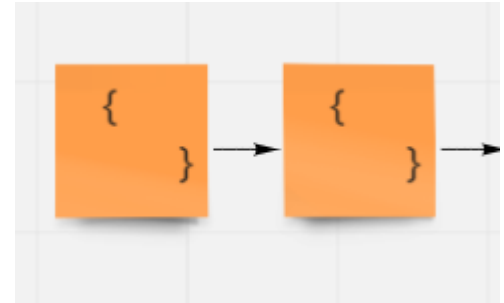
- Should be add only
 - 10l red wine ordered
 - 10l red wine order cancelled
 - 1l red wine ordered

no removing or adjusting!



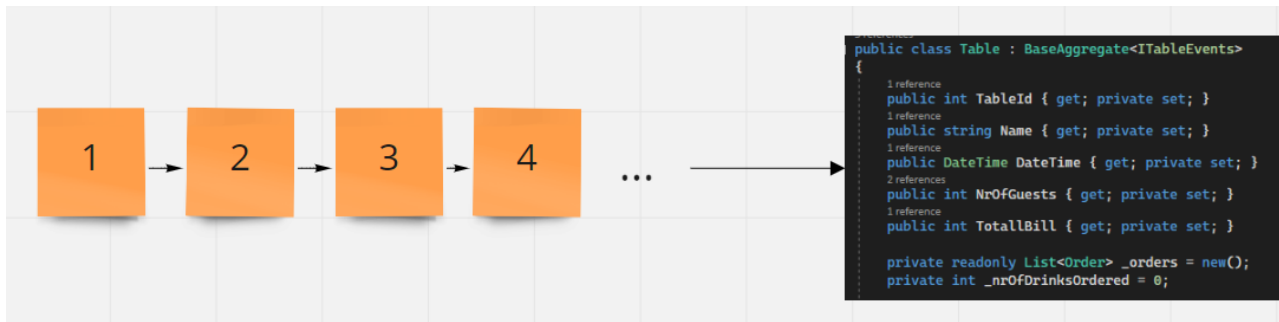
What if we now used our logs as "the only source of truth"

- use structured data such as a document store or json document



What if we now used our logs as "the only source of truth"

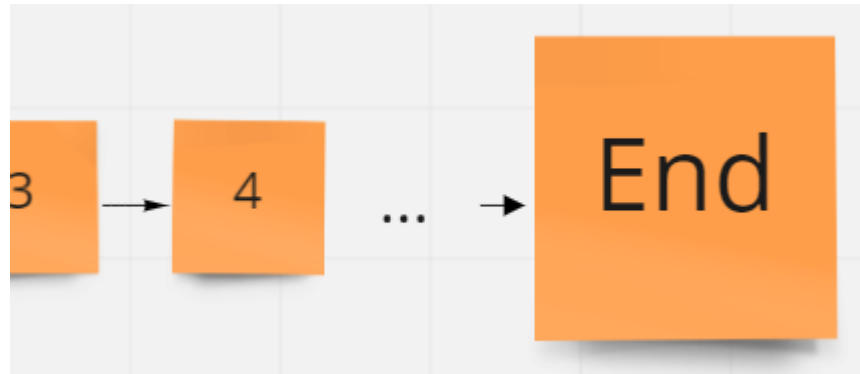
- Now you have time in your model and the sum of all transitions is the current state



What if we now used our logs as "the only source of truth"

When stream ends, mark it with an event

- Bill paid
- Table left



This is what we call "Event sourcing"

Instead of a table with rows for entities.
Use a stream for each aggregate.



```

_id: ObjectId('6834a692e035a983ea75eabe')
event: Object
metadata: Object
  eventName: "TableReserved"
  CurrentDateTime: 2025-05-26T17:36:18.408+00:00
  ReservationId: "5c2f98e6-1dfd-4972-a7f0-4b0d04a5f475"
  
```

```

_id: ObjectId('6834a696f034a5856b5faf9f')
event: Object
metadata: Object
  eventName: "DrinksOrdered"
  CurrentDateTime: 2025-05-26T17:36:22.695+00:00
  ReservationId: "5c2f98e6-1dfd-4972-a7f0-4b0d04a5f475"
  
```

```

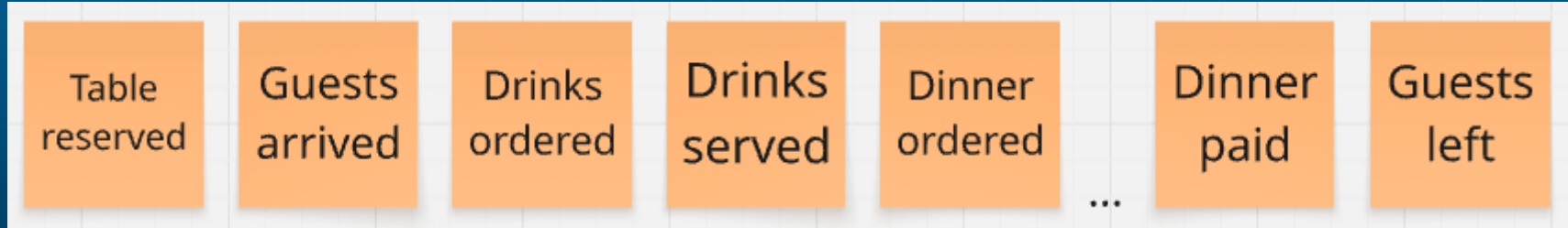
_id: ObjectId('6834bdf6f76696d374f22b9d')
event: Object
metadata: Object
  eventName: "DrinksServed"
  CurrentDateTime: 2025-05-26T19:16:06.773+00:00
  ReservationId: "5c2f98e6-1dfd-4972-a7f0-4b0d04a5f475"
  
```

This is what we call “Event sourcing”

Domain events must always be (re)playable.

It should always be possible to replay your domain event.
You obtain this by having it played immediately after you have completed your command. Then save the event to the db(stream).

Example in db(stream)



Where to begin?

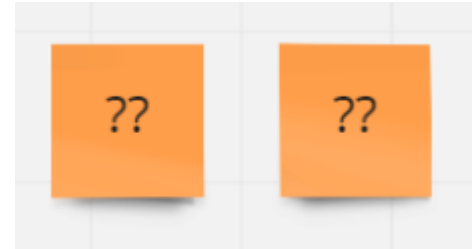


To explore your domain.



Event Storming

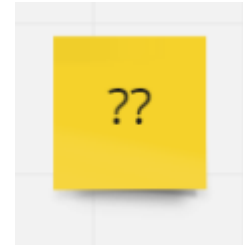
Because your events have yet to be determined and discovered.



Event Storming

Determine your aggregate.

=> your root object within your bounded context



Event Storming

Bridge between your developers and stakeholders.

Share the same knowledge and language.

Different stakeholders also have different needs and insights.

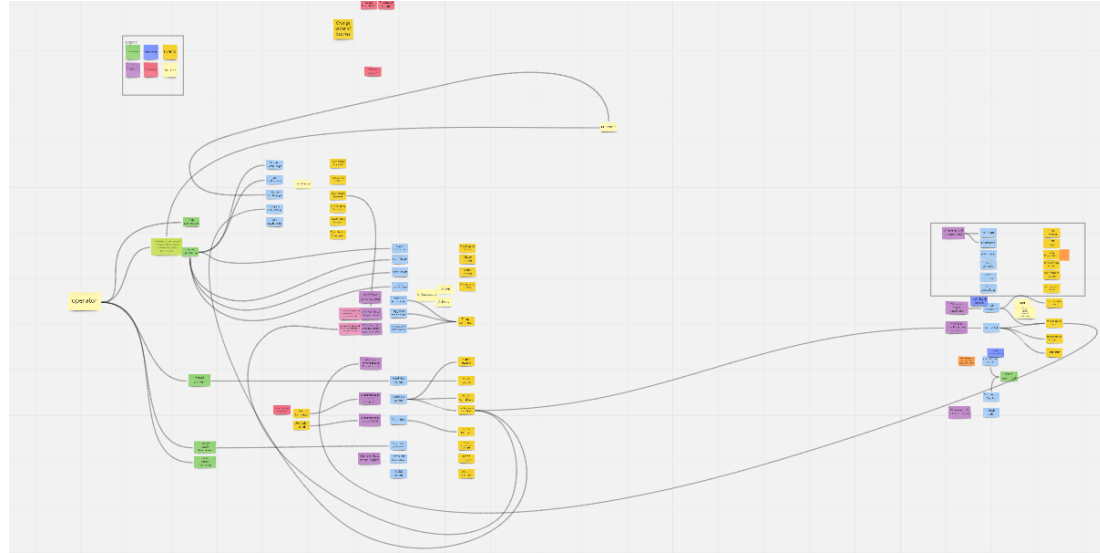
=> This can result into different bounded contexts.



Event Storming

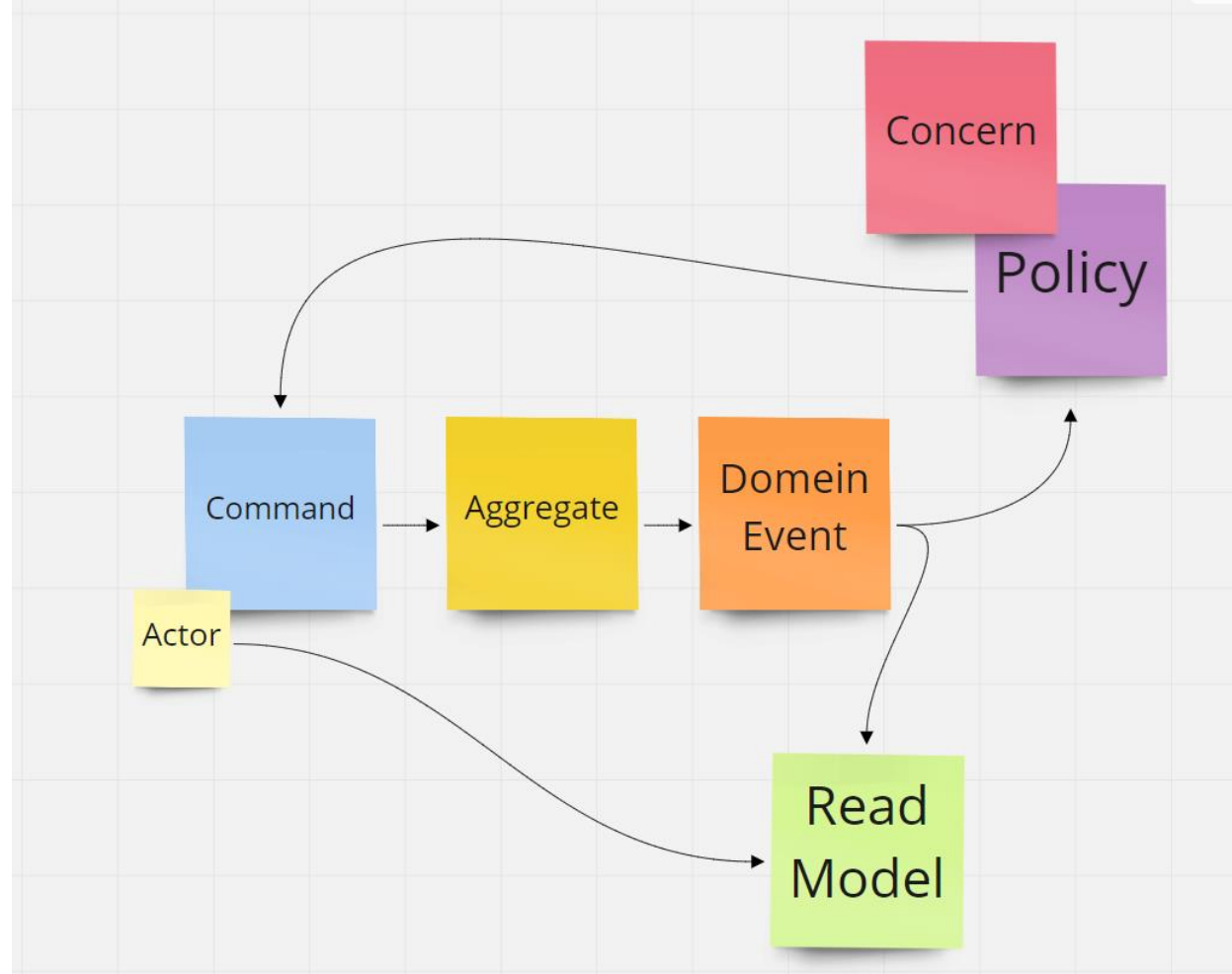
To better understand your domain.

This way you gain insights into the processes and phases of your application



Event Storming

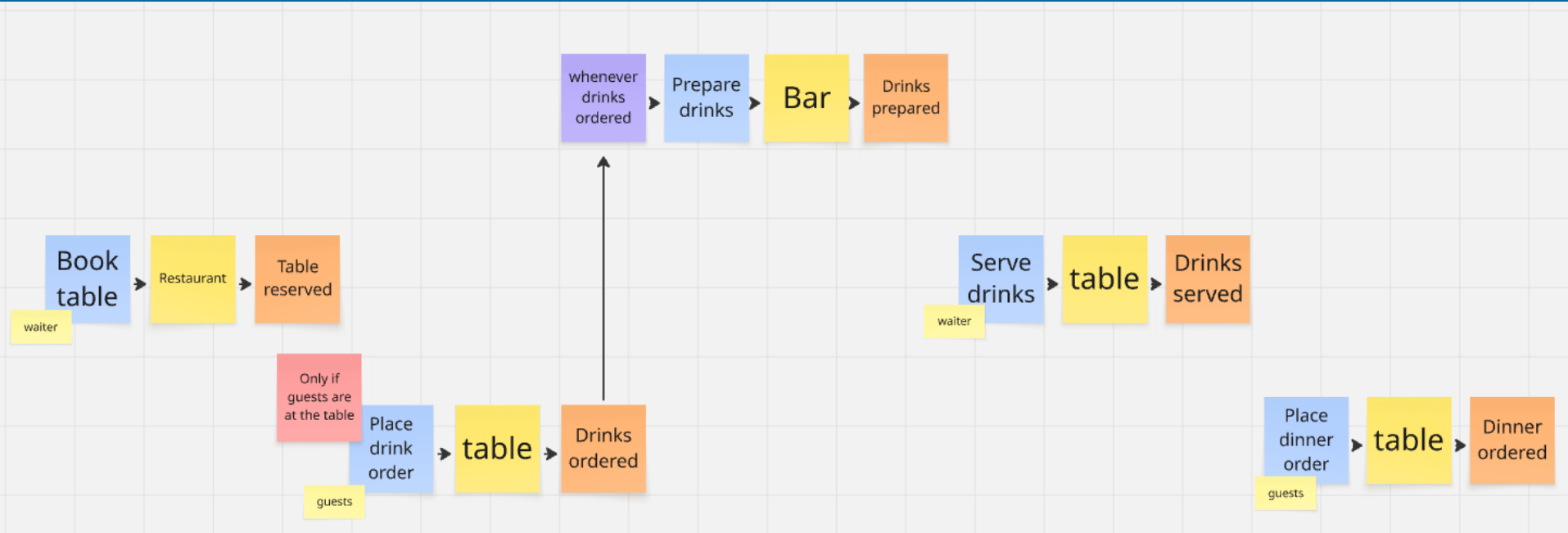
Process



Event Storming

- Is about interactions with the users/business.
- Domain Exploration.
- Using the language of the business.
- Interactions over time.
- System transformations.
- You cannot pay before you have ordered.

Example



Event Storming

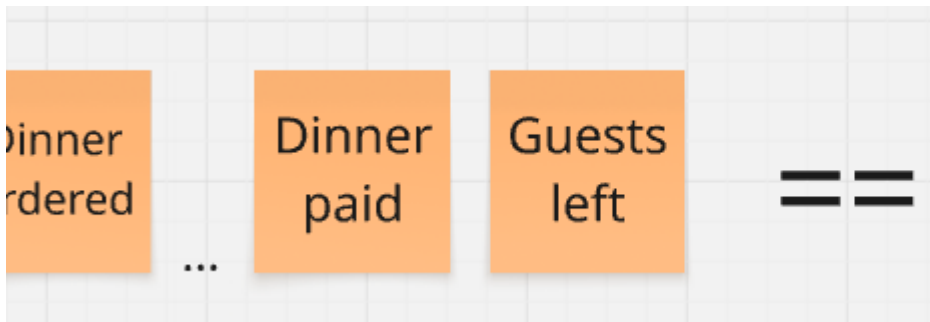
- Remains a continuous process with every change.
- It keeps improving your process from your model.

Event Storming

- Set up the space
 - Prepare a large wall or online board, sticky notes, and markers.
- Collect domain events
 - Let participants write and place events (past tense) on the timeline.
- Organize and group
 - Remove duplicates, sequence events, and group them by flow or theme.
- Add supporting elements
 - Use colored notes for commands (blue), actors/systems (yellow), and views (green).
- Explore and refine
 - Identify issues, gaps, and possible bounded contexts through discussion.

Event Storming

Let your code also reflect your model. Use the same names and format. This makes it easier to see the whole.



```
//play the event
1 reference
private void DrinksOrderedHandler(DrinksOrdered @event)
{
    _orders.Add(@event.Order);
    _nrOfDrinksOrdered += @event.Order.Quantity;
    TotalBill += @event.Order.Price;
}

1 reference
private void TableReservedHandler(TableReserved @event)
{
    ReservationId = @event.ReservationId;
    TableId = @event.TableId;
    Name = @event.Name;
    DateTime = @event.DateTime;
    NrOfGuests = @event.NrOfGuests;
}

1 reference
private void DrinksServedHandler(DrinksServed served)
{
    var existingOrder = _orders.FirstOrDefault(o :Order => o.OrderId == served.Order);
    _orders.Remove(existingOrder);
}
```

Constructors and Commands

```
2 references
public Table(int tableId, DateTime dateTime, string name, int nrOfGuests)
{
    RegisterHandlers();
    PublishNewEvent(new TableReserved(ReservationId: Guid.NewGuid() , tableId, name, dateTime, nrOfGuests));
}
```

```
3 references
public Table(IEnumerable<TableEvent> events)
{
    RegisterHandlers();

    //play all the evens
    events // IEnumerable<TableEvent>
        .ToList() // List<TableEvent>
        .ForEach(PlayEvent);
}
```

//command

```
3 references
public void OrderDrinks(Order order)
{
```

```
    if (_nrOfDrinksOrdered >= 2 * NrOfGuests)
        throw new Exception("Too many drinks :-)");
```

```
    PublishNewEvent(new DrinksOrdered(order));
}
```

1 reference

```
public void ServeDrinks(Guid order)
{
```

```
    var existingOrder = _orders.FirstOrDefault(o :Order => o.OrderId == order);
```

```
    if (existingOrder == null)
```

```
    {
        throw new Exception("This order dos not exists.");
    }
```

```
    PublishNewEvent(new DrinksServed(order));
}
```

Events

```
7 references | - changes | -authors, -changes  
▼ public record TableReserved(  
    int TableId,  
    string Name,  
    DateTime DateTime,  
    int NrOfGuests  
) : ITableEvents;
```

```
8 references | - changes | -authors, -changes  
public record DrinksOrdered(Order Order) : ITableEvents;  
  
12 references | - changes | -authors, -changes  
▼ public record Order(  
    string ProductName,  
    int ProductId,  
    int Quantity,  
    int Price,  
    string Comment);
```

Handlers

Internal

```
// register all events handlers
2 references
private void RegisterHandlers()
{
    RegisterHandler<TableReserved>(TableReservedHandler);
    RegisterHandler<DrinksOrdered>(DrinksOrderedHandler);
    RegisterHandler<DrinksServed>(DrinksServedHandler);
}

//play the event
1 reference
private void DrinksOrderedHandler(DrinksOrdered @event)
{
    _orders.Add(@event.Order);
    _nrOfDrinksOrdered += @event.Order.Quantity;
    TotalBill += @event.Order.Price;
}

1 reference
private void TableReservedHandler(TableReserved @event)
{
    ReservationId = @event.ReservationId;
    TableId = @event.TableId;
    Name = @event.Name;
    DateTime = @event.DateTime;
    NrOfGuests = @event.NrOfGuests;
}

1 reference
private void DrinksServedHandler(DrinksServed served)
{
    var existingOrder = _orders.FirstOrDefault(o : Order => o.OrderId == served.Order);
    _orders.Remove(existingOrder);
}
```

At Work?

For each command, the aggregate gets in the correct state by loading all events from the current stream.

Then execute your commands and add your events to the current stream.

```
0 references
public class OrderDrinksCommandHandler(TablesStore tablesStore)
    : IRequestHandler<OrderDrinksCommand>
{
    0 references
    public async Task Handle(OrderDrinksCommand request, Cancellation
    {
        var table = await tablesStore.Get(request.ReservationId);

        table.OrderDrinks(request.Order);

        await tablesStore.SaveAsync(table);
    }
}
```

Testing?

[Fact]

0 references | 0 changes | 0 authors, 0 changes

```
public async ValueTask Should_Emit_TableReservedEvent_On_Create()
{
    var aggregate = new Table(6, new DateTime(year: 2021, month: 12, day: 31), name: "kim vr", nrOfGuests: 2);
    var events = new List<ITableEvents>();
    await aggregate.PlayAllEvents(writeEvent: async e:ITableEvents =>
    {
        //because normally this is an async write to the db
        await Task.FromResult(0);
        events.Add(e);
    }); // ValueTask

    events.Should().ContainSingle();

    events[0] // ITableEvents
        .Should() // ObjectAssertions
        .BeEquivalentTo(expectation: new TableReserved(
            TableId: 6,
            Name: "kim vr",
            DateTime: new DateTime(year: 2021, month: 12, day: 31),
            NrOfGuests: 2));
}
```

Testing?

```
public async ValueTask Should_only_allow_2_times_a_drinks_order()
{
    var events = new TableEvent[]
    {
        new TableReserved(...),
        new DrinksOrdered(...),
        new DrinksOrdered(...),
    };

    var newEvents = new List<TableEvent>();

    var aggregate = new Table(events);
    var act : Action = () => aggregate.OrderDrinks(
        new Order(
            Guid.NewGuid(),
            ProductName: "Martini",
            ProductId: 8,
            Quantity: 2,
            Price: 14,
            Comment: ""
        ));

    act.Should() // ActionAssertions
        .Throw<Exception>()
        .WithMessage(expectedWildcardPattern: "Too many drinks :-)");

    await aggregate.PlayAllEvents(writeEvent: async e : TableEvent =>
    {
        await Task.FromResult(0);
        newEvents.Add(e);
    }); // ValueTask

    newEvents.Any().Should().BeFalse();
}
```

```
public async ValueTask Should_be_able_to_order_a_drink()
{
    var events = new ITableEvents[]
    {
        new TableReserved(
            TableId: 6,
            Name: "kim vr",
            DateTime: new DateTime(year: 2021, month: 12, day: 31),
            NrOfGuests: 2
        )
    };

    var aggregate = new Table(events);
    aggregate.OrderDrinks(new Order(
        ProductName: "Martini",
        ProductId: 6,
        Quantity: 2,
        Price: 6,
        Comment: ""
    ));

    var newEvents = new List<ITableEvents>();
    await aggregate.PlayAllEvents(writeEvent: async e : ITableEvents =>
    {
        await Task.FromResult(0);
        newEvents.Add(e);
    }); // ValueTask

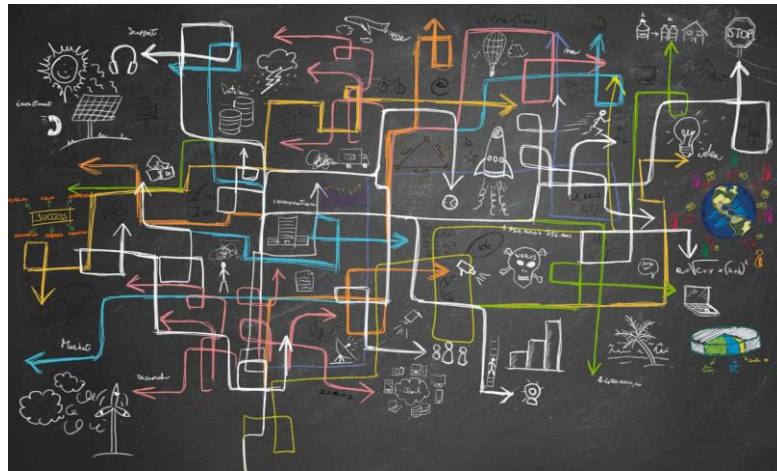
    newEvents.Should().ContainSingle();
    newEvents[0] // ITableEvents
        .Should() // ObjectAssertions
        .BeEquivalentTo(
            expectation: new DrinksOrdered(
                new Order(
                    ProductName: "Martini",
                    ProductId: 6,
                    Quantity: 2,
                    Price: 6,
                    Comment: ""
                )
            )
        );
}
```


When to use Event Sourcing?



When to use Event Sourcing?

If you have many complex transitions and interactions.



When to use Event Sourcing?

If you need audits. Or must be able to prove how you arrived at a state.



When to use Event Sourcing?

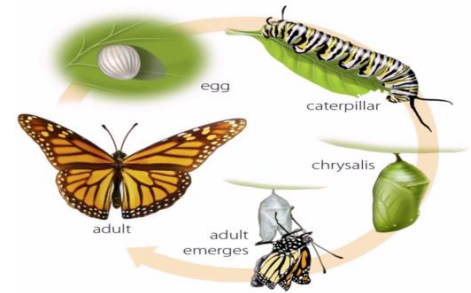
If you want to be able to debug what happened in production.



When to use Event Sourcing?

If you need time in your model.

=> For example: if first a and then c happens then x does something. If first b and then a happens then x is not allowed or does something completely different.

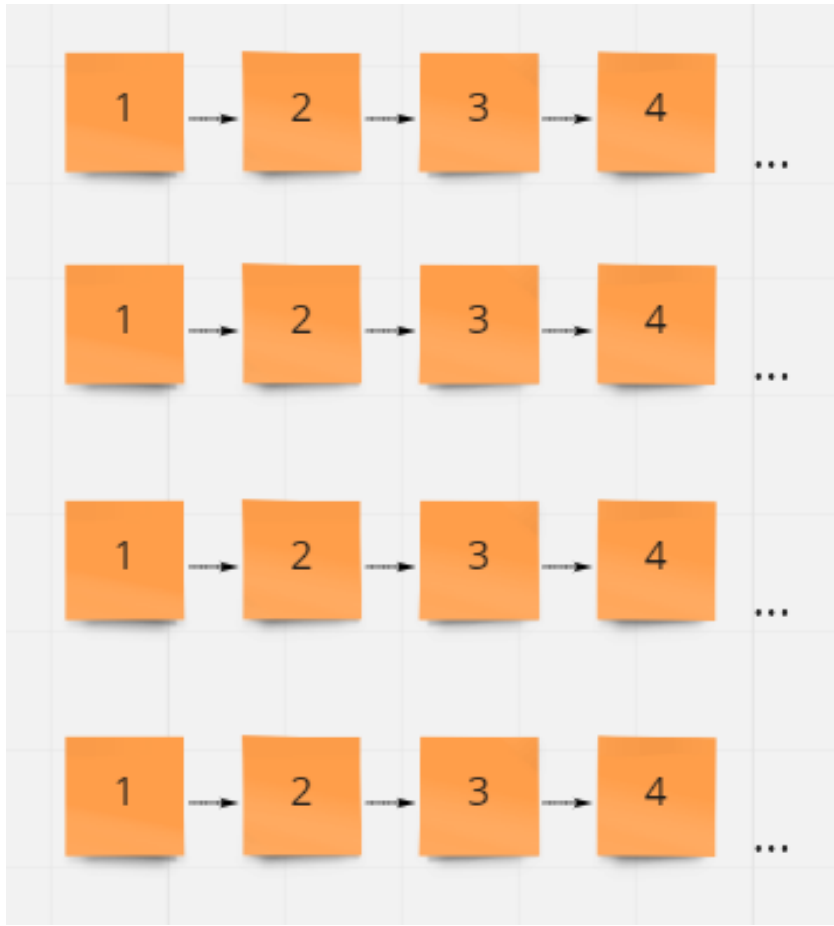


Pro and contra

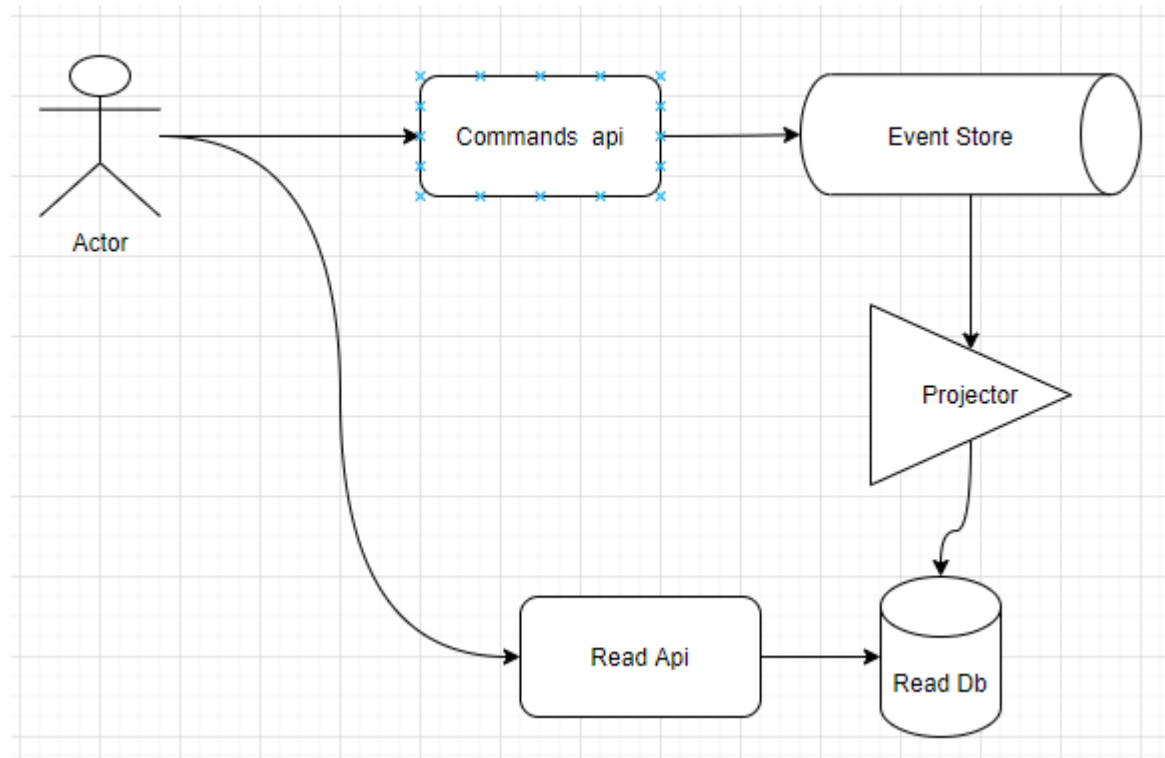
- To have time in your system
 - To query over time
 - To move your complexity in the aggregates
 - **When mastered**, your system becomes modular
- It requires a lot of discipline from the team.
 - Your system becomes more complex
 - Your system feels less familiar



Select * from Tables where firstOrder != appetizer and ...

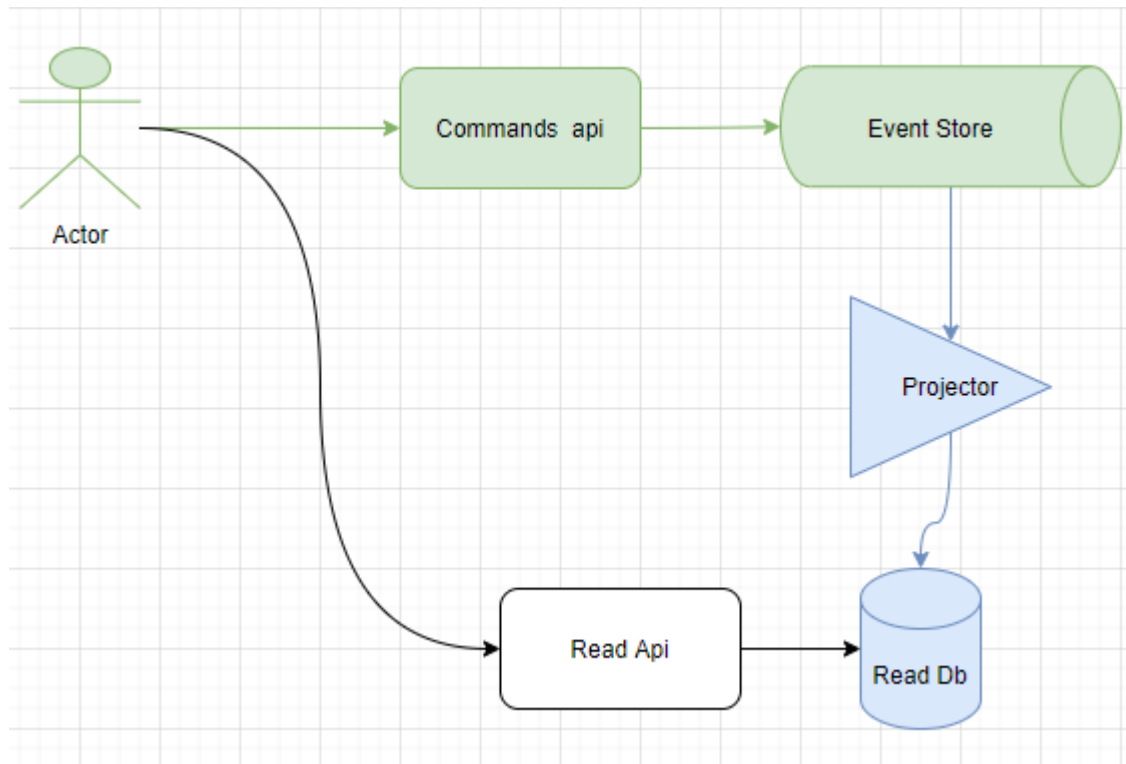


Command Query Responsibility Segregation



CQRS

What do we already have



CQRS

Where to use a read model?

Everything you need to visualize

- A list
- A detail e.g. all reserved tables
- Everything that is processed in the kitchen
- What are the three most common orders
- What is the average price per minute and what are the top 5 orders



CQRS

Every domain event that is needed for your read model is translated into a public event

```
8 references | - changes | -authors, -changes  
public record DrinksOrdered(Order Order) : ITableEvents;
```

```
namespace EventSourcingDemo.PublicEvents;
```

```
7 references
```

```
public record DrinksOrdered(Guid OrderId, Guid ReservationId, Order Order, int TableId, string Name) : INotification;
```

CQRS

Corresponding records are retrieved adjusted and stored in the DB



CQRS

Events should no longer be ordered.

1. If you received a debit of €100 for a credit card once.
2. You can't find the credit card number.
3. Then just create it with an amount of €0.
4. Debit the card for €100, where it turns negative.
5. If then you create comes in for €200. then it will end at €100.

CQRS

Advantage?

- To deploy your read and write db separately.
- Scale separately.
 - Filing taxes at the last minute will cause much less inconvenience.
- If there are many transactions, your read db will catch up slowly. But it doesn't block your command side.
- To create a completely new read model and feed it with all historical event.
 - New report contains all historical data after processing.

**Those same public events can also be sent to other services.
There they are processed by policies.**



Code @

<https://github.com/kimVanRenterghemNew/EventSourcingAndCQRS>

 kimVanRenterghemNew cleanup	f56fb49 · 45 minutes ago	🕒 12 Commits
📁 EventSourcingDemo.Api	cleanup	45 minutes ago
📁 EventSourcingDemo.Application	cleanup	45 minutes ago
📁 EventSourcingDemo.Integration.Test	cleanup	45 minutes ago
📁 EventSourcingDemo.MongoDb	cleanup	45 minutes ago
📁 EventSourcingDemo.Test	cleanup	45 minutes ago
📁 EventSourcingDemo	cleanup	45 minutes ago
📄 .editorconfig	dotnet 8	11 months ago
📄 .gitignore	demo code for visugxl-2021	4 years ago
📄 EventSourcingDemo.In	add orders list for kitchen and serve order	1 hour ago

Demo

Questions?