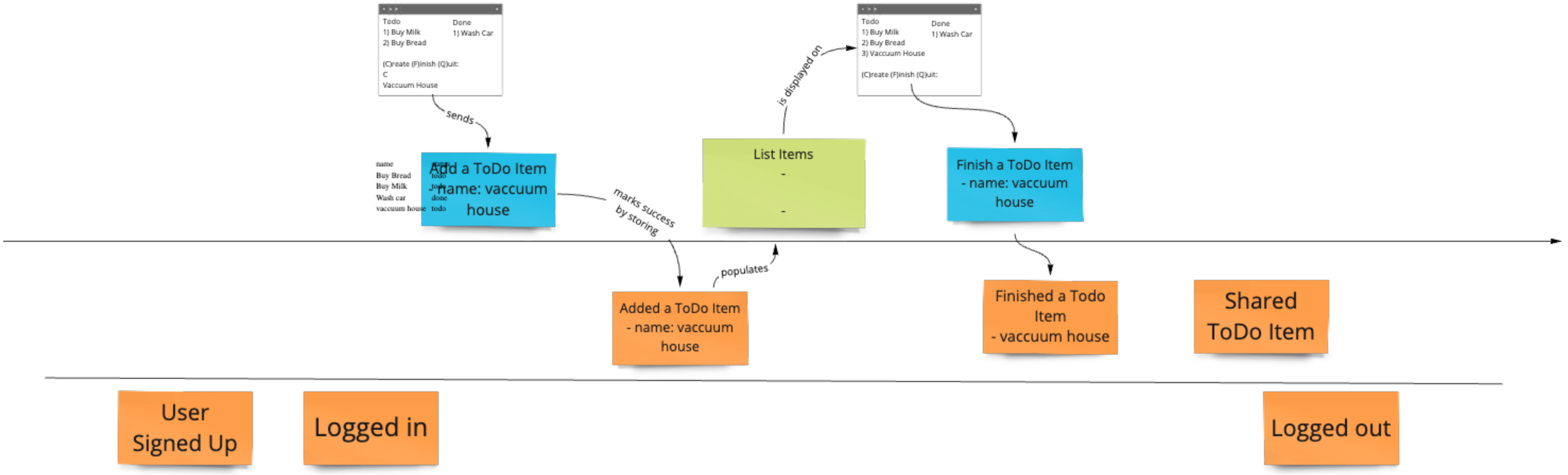
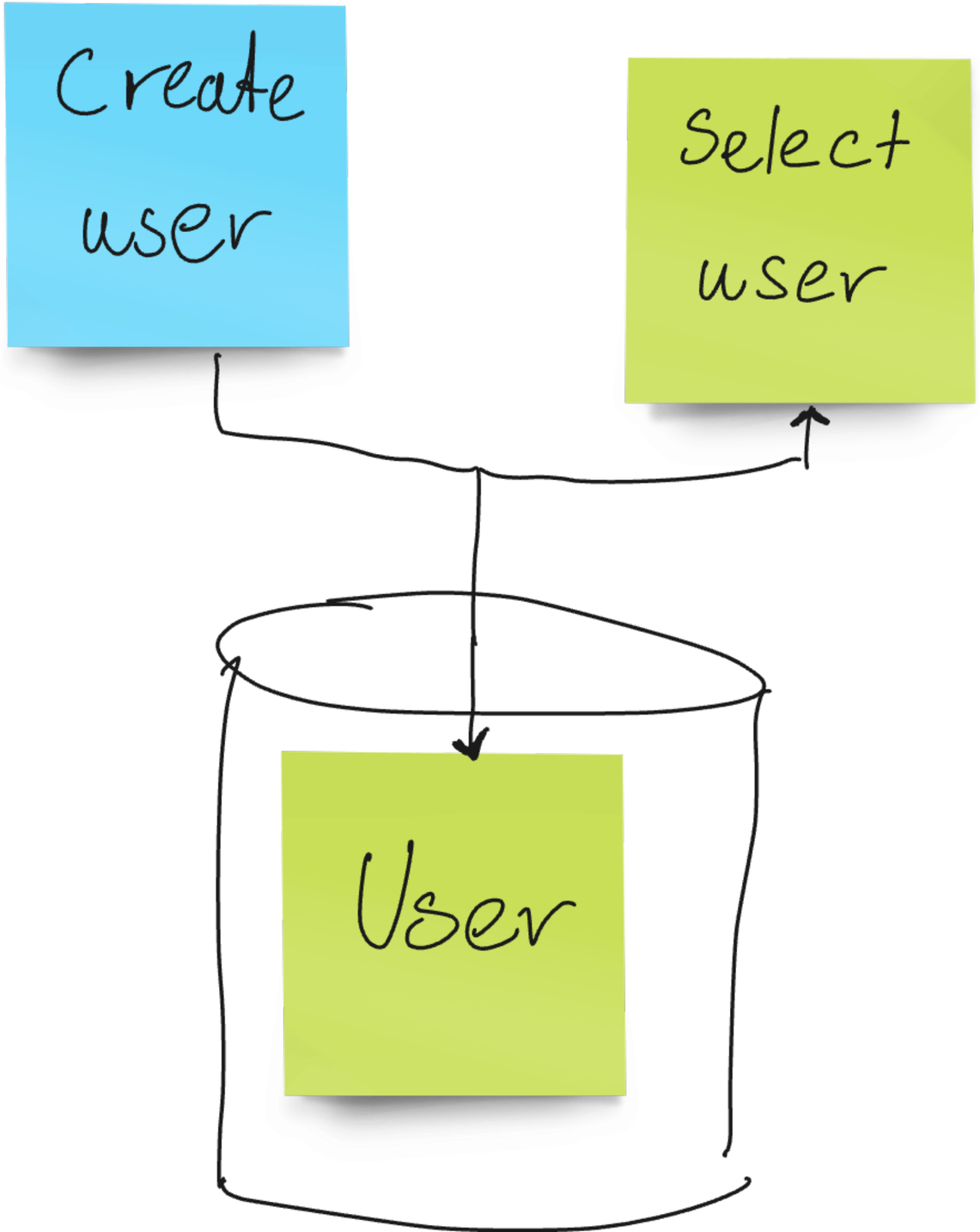
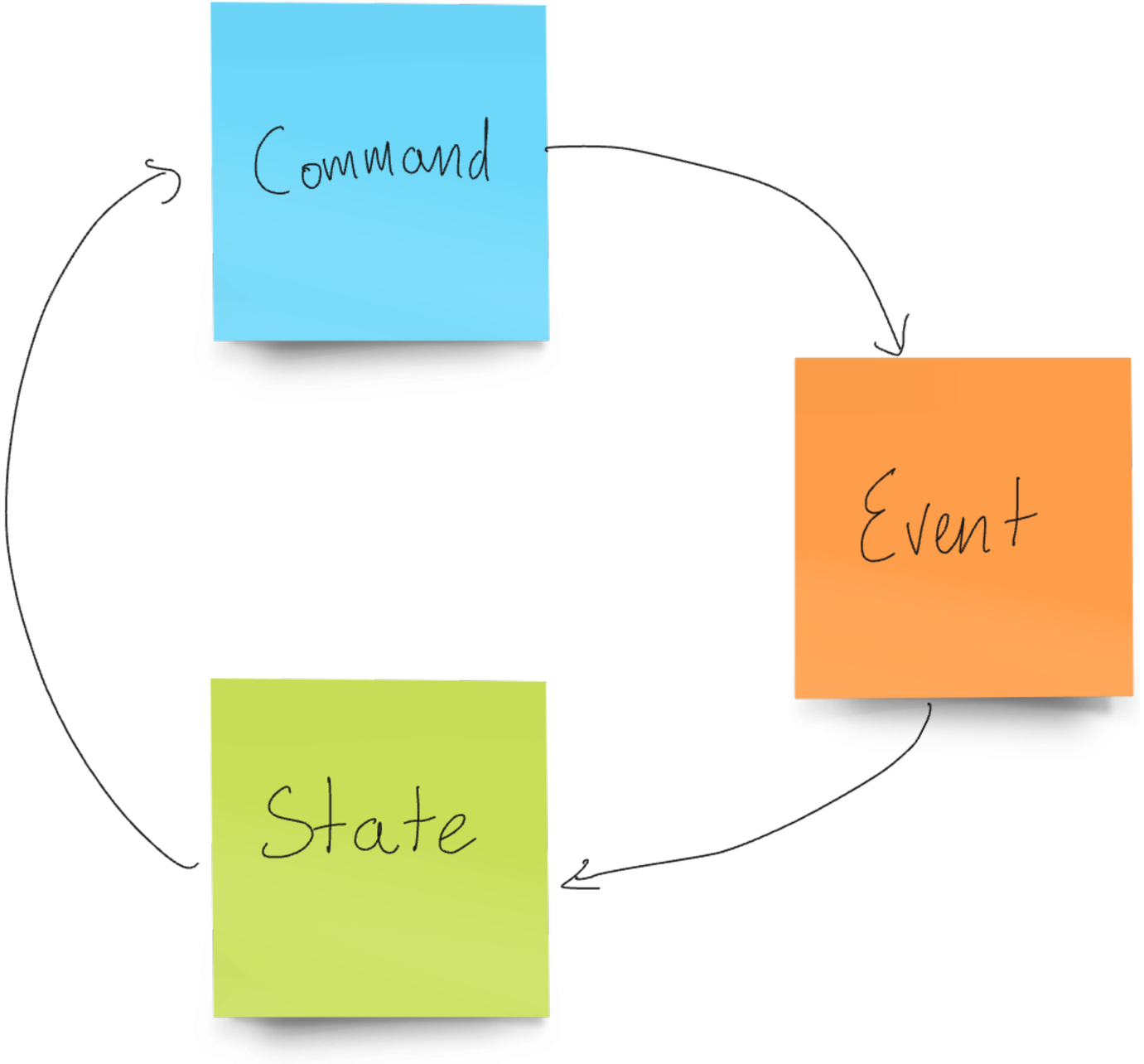
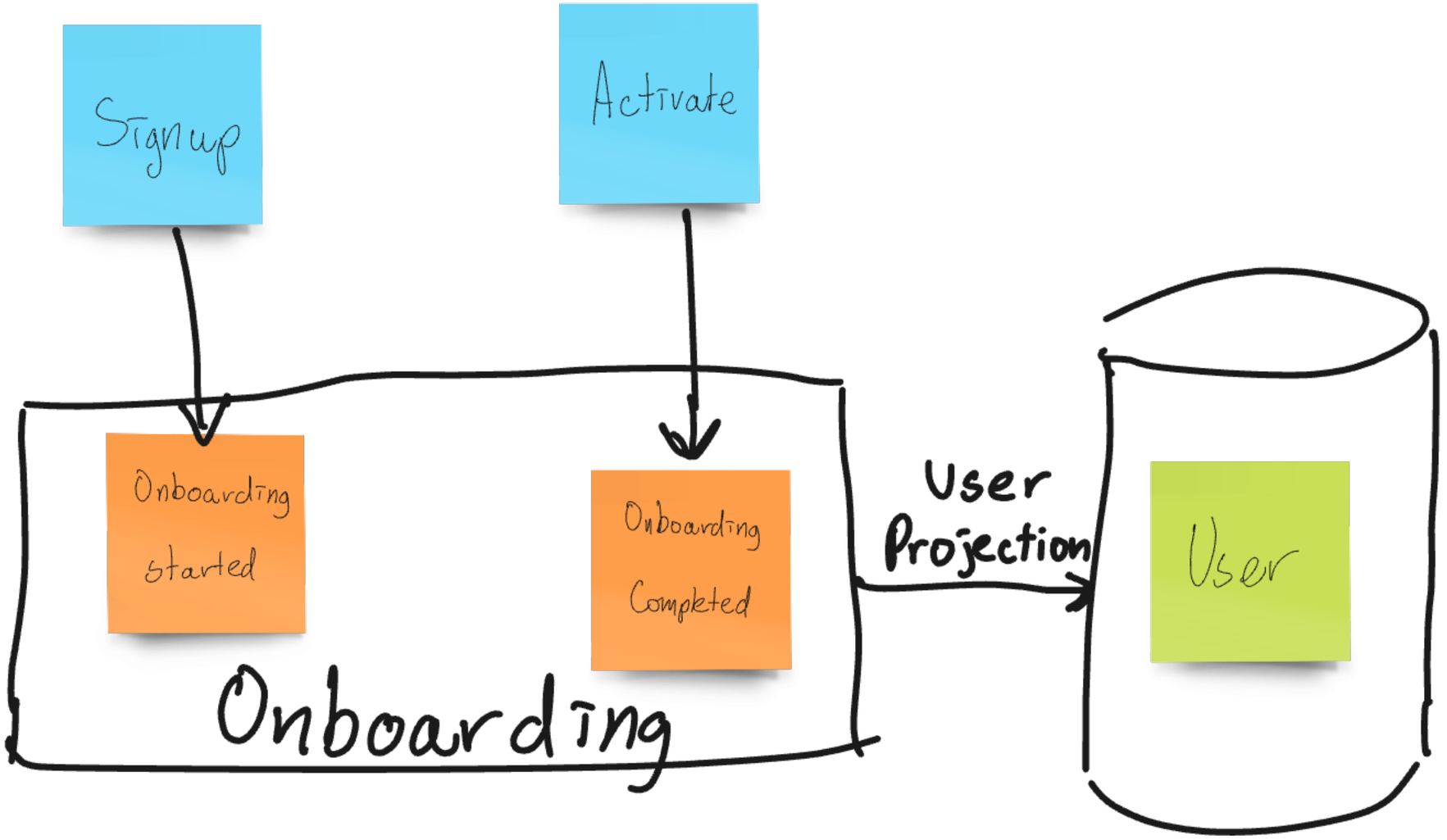

Building
Reactive apps
one event
@
a time

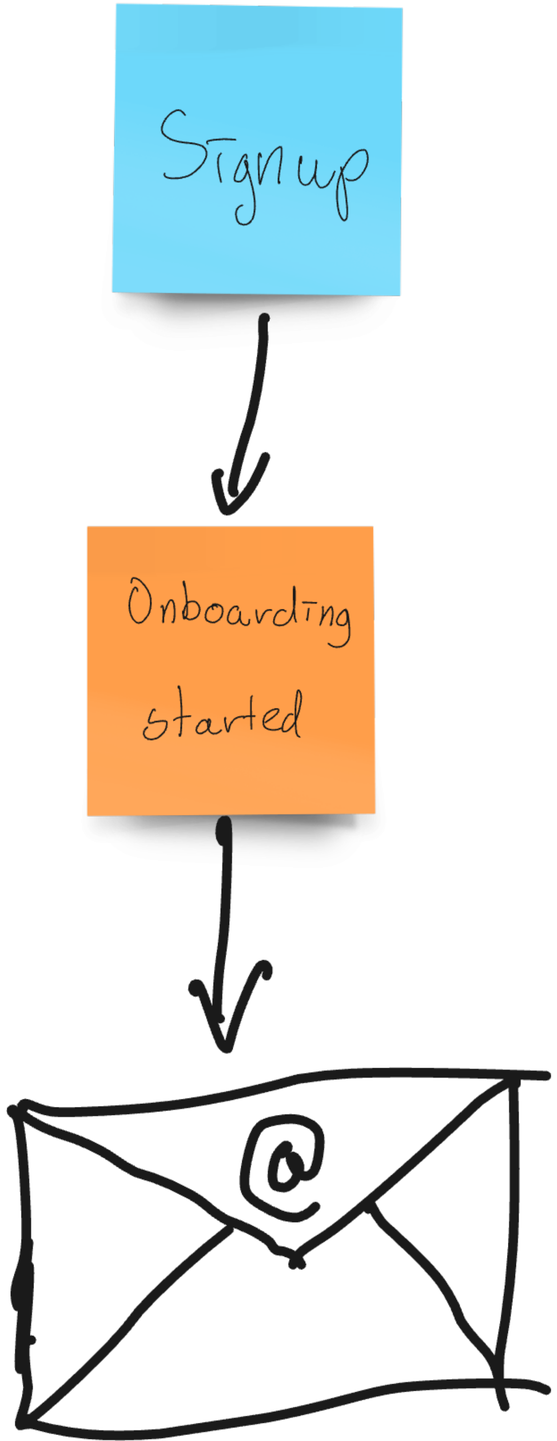
EVENTMODELING.org

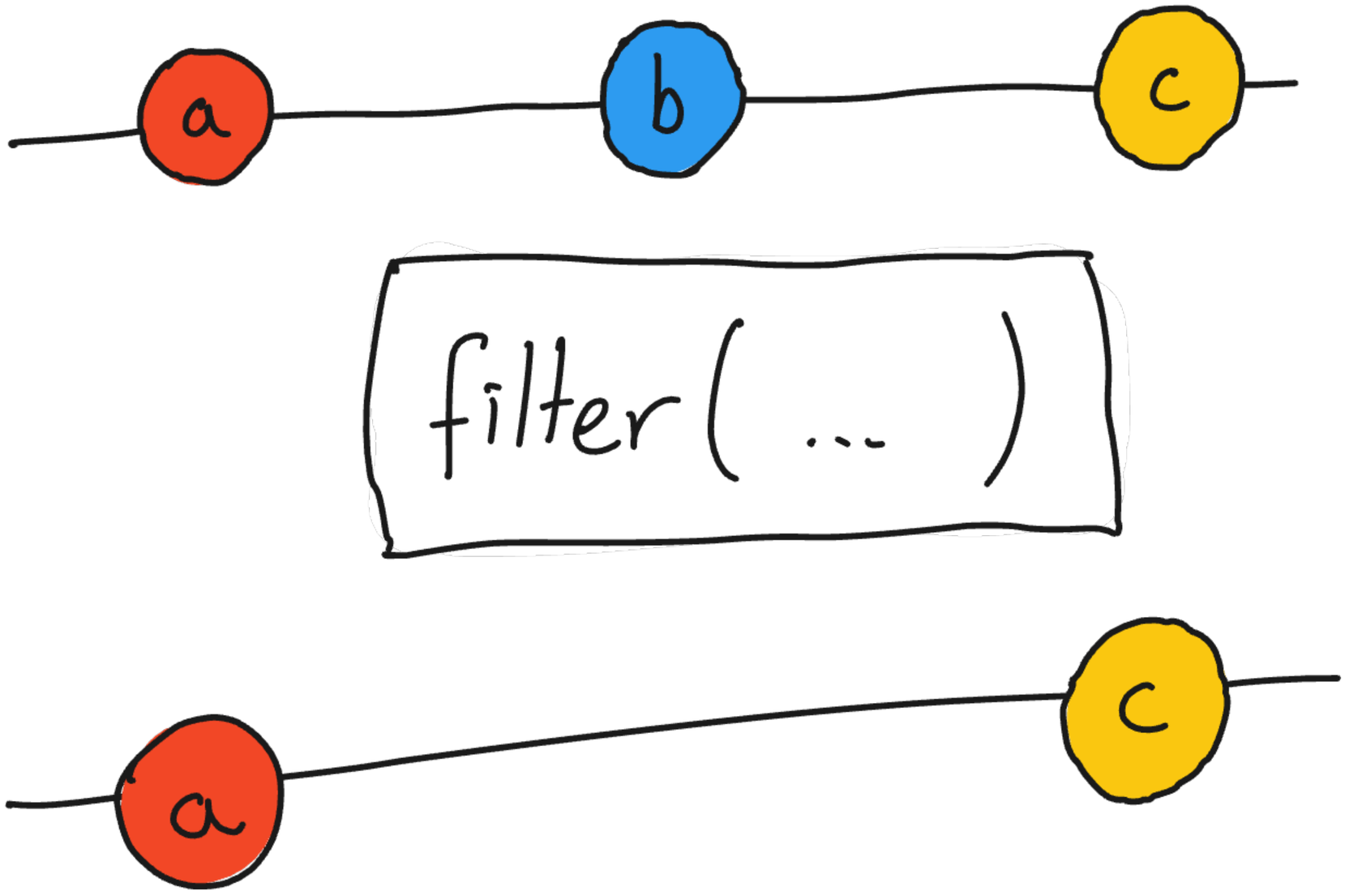


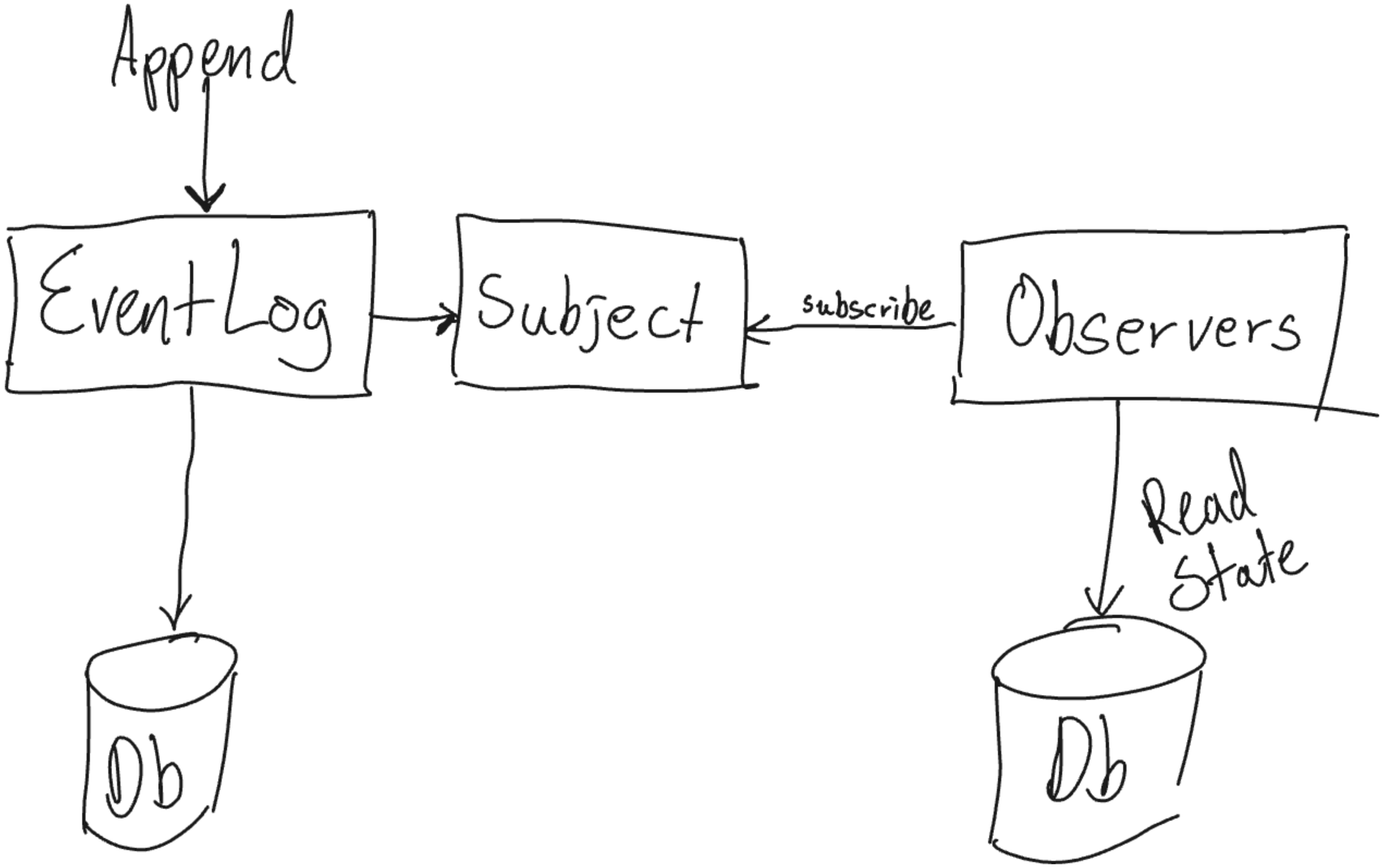

Create user
Select user
User


Command
Event
State


Signup
Activate
Onboarding started
Onboarding Compkted
Onboarding
User Projection
User


Signup
Onboarding
started


a
b
c
filter(...)
a
c


Append
EventLog
Subject
subscribe
Observers
Db
Read
State
Db

Oskar Dudycz • 1st
Event Sourcerer / Helping others build Event-Driven systems
1d •

I had such thought today: in Event Sourcing Events are Facts and Read Models are Rumours 😎

49

15 comments • 2 reposts

https://www.eventstore.com

https://martendb.io

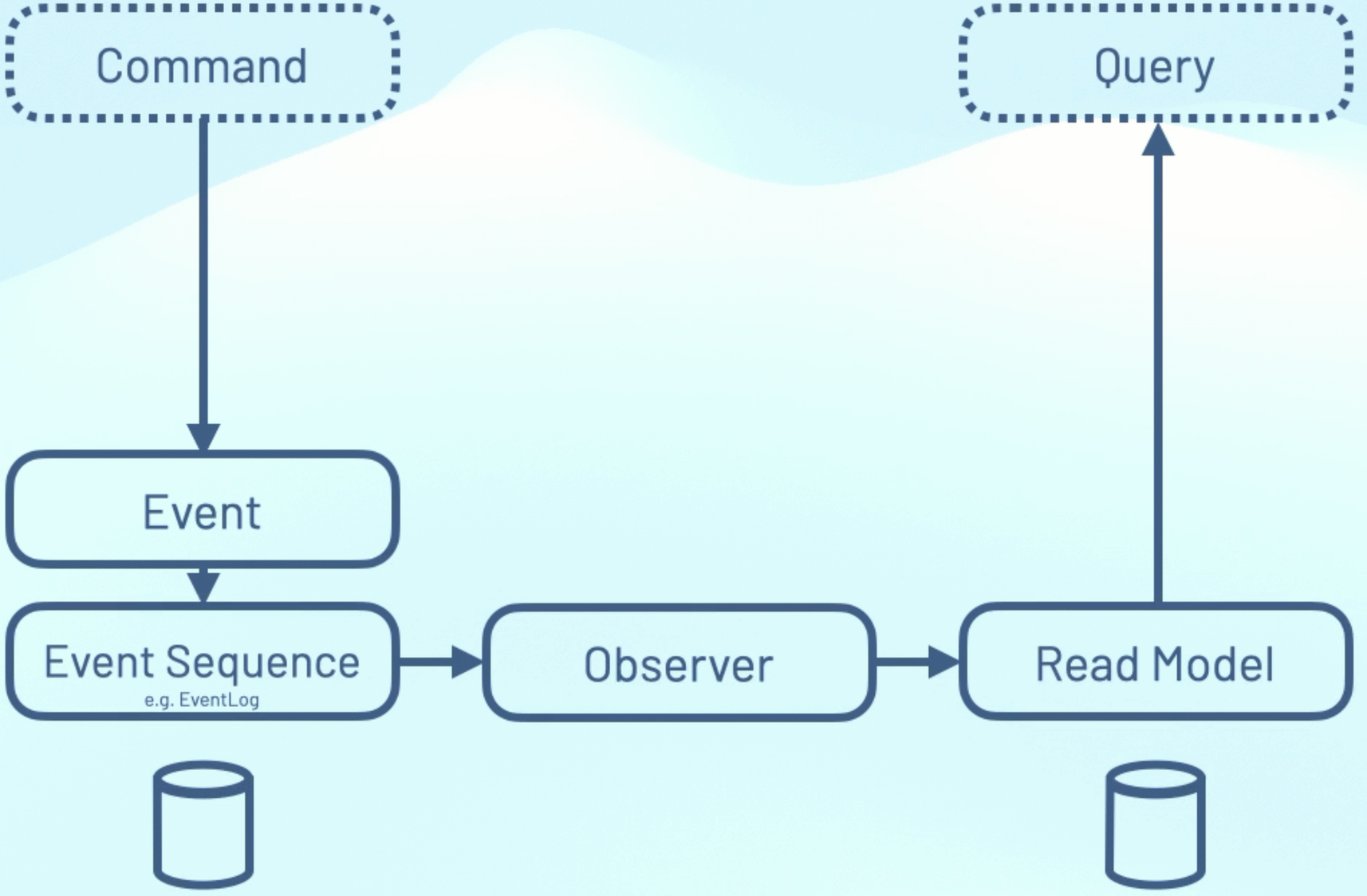
https://eventuous.dev

Emmet

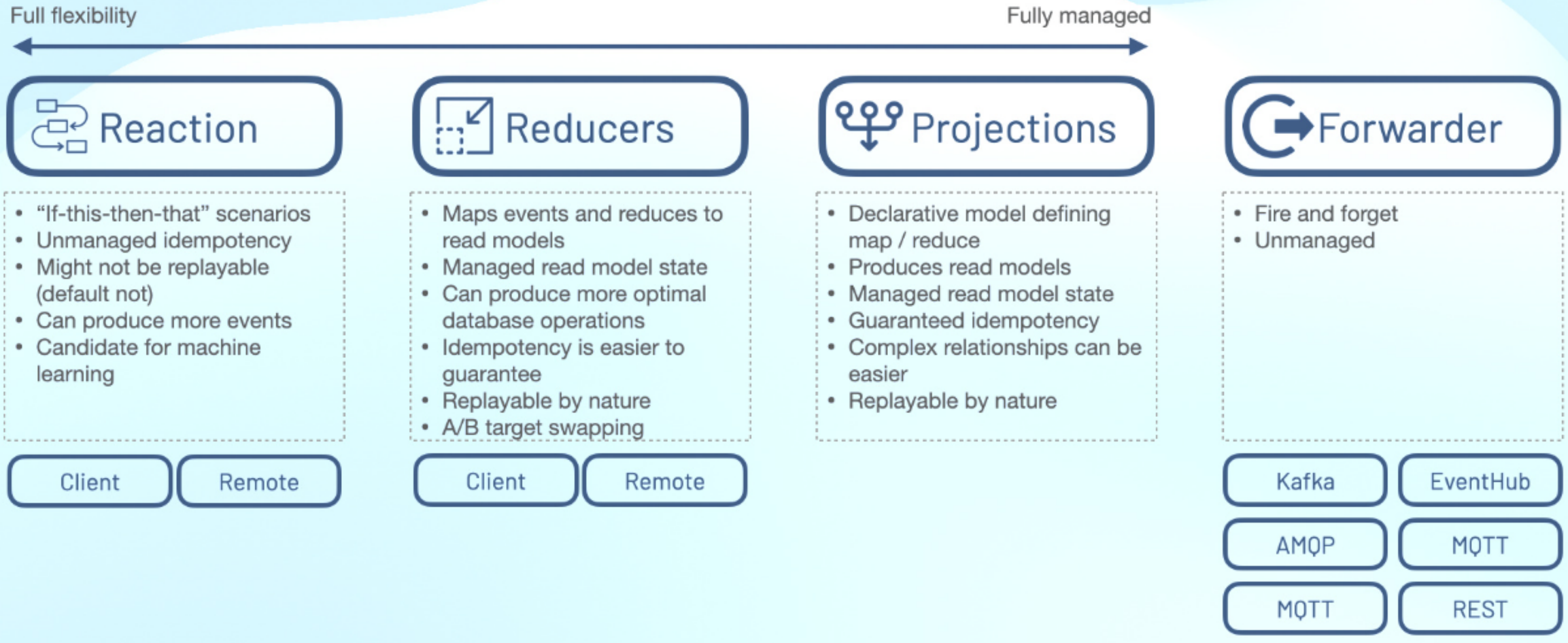
https://github.com/event-driven-io/emmett

https://cratis.io

Pipelines


Command
Query
Event
Event Sequence
e.g. EventLog
Observer
Read Model

Observer types



```
public record ApplicationEnvironmentProjection : IProjectionFor<ApplicationEnvironment>
{
    0 references
    public ProjectionId Identifier => "27449588-0f8c-47e4-b1ee-c326a12ed391";

    0 references
    public void Define(IProjectionBuilderFor<ApplicationEnvironment> builder) => builder
        .From<ApplicationEnvironmentCreated>(_ => _
            .UsingCompositeKey<ApplicationEnvironmentKey>(_ => _
                .Set(k => k.ApplicationId).To(e => e.ApplicationId)
                .Set(k => k.EnvironmentId).ToEventSourceId())
            .Set(m => m.Name).To(e => e.Name)
            .Set(m => m.DisplayName).To(e => e.DisplayName)
            .Set(m => m.ShortName).To(e => e.ShortName)
            .Set(m => m.CratisVersion).To(e => e.CratisVersion));
}
```

```
public void Define(IProjectionBuilderFor<ApplicationEnvironmentsForApplication> builder) => builder
    .Children(m => m.Environments, cb => cb
        .IdentifiedBy(m => m.Id)
        .From<ApplicationEnvironmentAddedToApplication>(e => e
            .UsingKey(e => e.EnvironmentId)));
```

```
public void Define(IProjectionBuilderFor<ApplicationHierarchyForListing> builder) => builder
    .From<ApplicationCreated>(_ => _
        .Set(m => m.Name).To(e => e.Name))
    .Children(_ => _.Environments, _ => _
        .FromEvery(_ => _
            .Set(m => m.LastUpdated).ToEventContextProperty(c => c.Occurred)
            .IncludeChildProjections())
        .IdentifiedBy(m => m.EnvironmentId)
```

```
public class OpenDebitAccountRules : RulesFor<OpenDebitAccountRules, OpenDebitAccount>
{
    0 references
    public override RuleId Identifier => "9c09c285-0eea-4632-ac2d-0d23c7ac10ba";

    0 references
    public IEnumerable<AccountName> Accounts { get; set; } = Array.Empty<AccountName>();

    0 references
    public OpenDebitAccountRules()
    {
        RuleForState(_ => _.Accounts)
            .Unique(_ => _.Details.Name)
            .WithMessage("Account with name already exists");
    }

    0 references
    public override void DefineState(IProjectionBuilderFor<OpenDebitAccountRules> builder) => builder
        .Children(_ => _.Accounts, _ => _
            .IdentifiedBy(_ => _)
            .From<DebitAccountOpened>(_ => _.UsingKey(_ => _.Name)));
}
```

```
var eventSourceId = (EventSourceId)"299681c4-f100-4dea-bfea-633115349ed1";
var order = await aggregateRootFactory.Get<Order>(eventSourceId);
order.DoStuff();
order.DoOtherStuff();
await order.Commit();
```

```
public class Order : AggregateRoot
{
    2 references
    readonly List<CartItem> _cartItems = new();

    0 references
    void On(ItemAddedToCart @event)
    {
        _cartItems.Add(new CartItem(@event.MaterialId, @event.Quantity));
    }

    0 references
    void On(ItemRemovedFromCart @event, EventContext context)
    {
        _cartItems.RemoveAll(_ => _.MaterialId == @event.MaterialId);
    }
}
```

```
public class Order : AggregateRoot<OrderState>
{
}

7 references
public record OrderState(int Items, IEnumerable<CartItem> CartItems);

You, 1 second ago | 1 author (You)
[Reducer("5027a520-6d25-47c7-9d52-b1e9f82905d2")]
0 references
public class OrderStateReducer : IReducerFor<OrderState>
{
    0 references
    public Task<OrderState> ItemAdded(ItemAddedToCart @event, OrderState? initial, EventContext context)
    {
        initial ??= new OrderState(0, Enumerable.Empty<CartItem>());
        initial = initial with { Items = initial.Items + 1 };
        return Task.FromResult(initial);
    }
}
```

```
public class Order : AggregateRoot<OrderState>
{
}

7 references
public record OrderState(int Items, IEnumerable<CartItem> CartItems);

0 references | You, 6 days ago | 1 author (You)
public class OrderStateProjection : IImmediateProjectionFor<OrderState>
{
    0 references
    public ProjectionId Identifier => "4c6f7eac-d74d-425b-b2fd-e32e8e365b32";

    0 references
    public void Define(IProjectionBuilderFor<OrderState> builder) => builder
        .Children(_ => _.CartItems, cb => cb
            .IdentifiedBy(m => m.MaterialId)
            .From<ItemAddedToCart>(_ => _
                .UsingKey(e => e.MaterialId)
                .Set(m => m.Quantity).To(e => e.Quantity))
            .RemovedWith<ItemRemovedFromCart>()
            .From<QuantityAdjustedForItemInCart>(_ => _
                .UsingKey(e => e.MaterialId)
                .Set(m => m.Quantity).To(e => e.Quantity)));
}
```

1ST EDITION

Metaprogramming in C#

Automate your .NET development and simplify overcomplicated code

EINAR INGEBRIGTSEN

Stop

losing data!!

KONGSBERG

Einar Ingebrigtsen

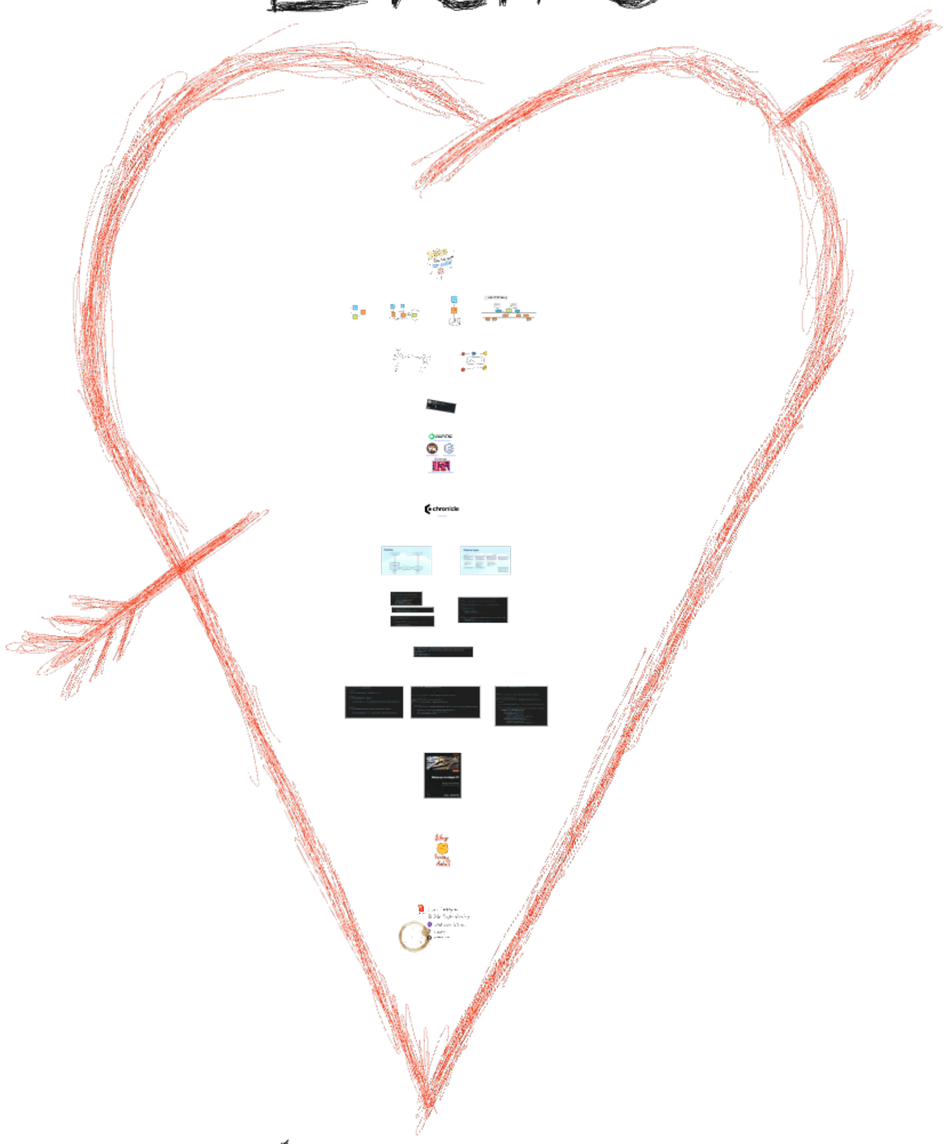
https://ingebrigtsen.blog

linkedin.com/in/einari

x.com/einari

github.com/einari

Events



4evva