Durable **messaging** and related patterns, for fun and profit.

Ismael Celis
ismaelcelis.com

Durable **messaging** and related patterns, for fun and profit.

(But really, just fun)

Ismael Celis
ismaelcelis.com

Jun 27, 2022 · Ismael Celis · 5 minutes read

Event Sourcing from the ground up, with Ruby examples, part 1

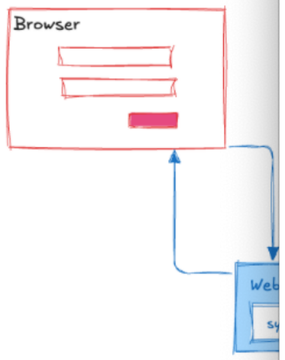
ismaelcelis.com/posts/2025-11-un

Nov 4, 2025 · Ismael Celis · 6 minutes read

Unfinished bus

The conventional distinction betw
"background" operations in web c
incomplete.

There, I said it.



Conventional browse

Big MVC™ wants us to believe tha
synchronous, ie. it runs, changes

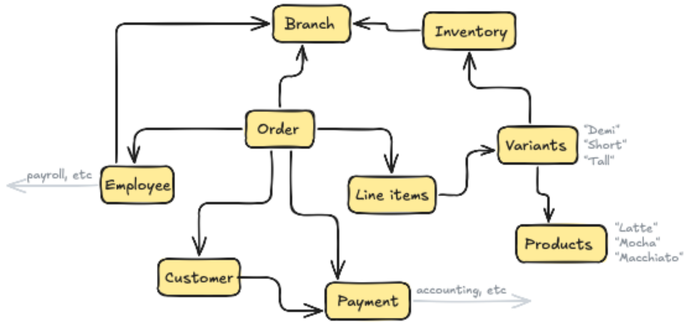
ismaelcelis.com/posts/2025

Apr 21, 2025 · Ismael Celis · 9 minutes read

Give it time

Our experience of reality is based on the passage of time. Things begin and end, events happen one after the other. The world *changes*. And yet we model software as static object graphs frozen in time.

This sort of makes sense. While I'm used to going through my day as events in time (wake up, have breakfast, shower, *really* wake up, etc), the act of *understanding* feels like building a graph. When I plan for a new project, my mind jumps to try and form a picture of the current structure of the system. I try to understand how the different components interact with each other, and how they relate to the business domain. I think in hierarchies.



A relational data model for a coffee shop

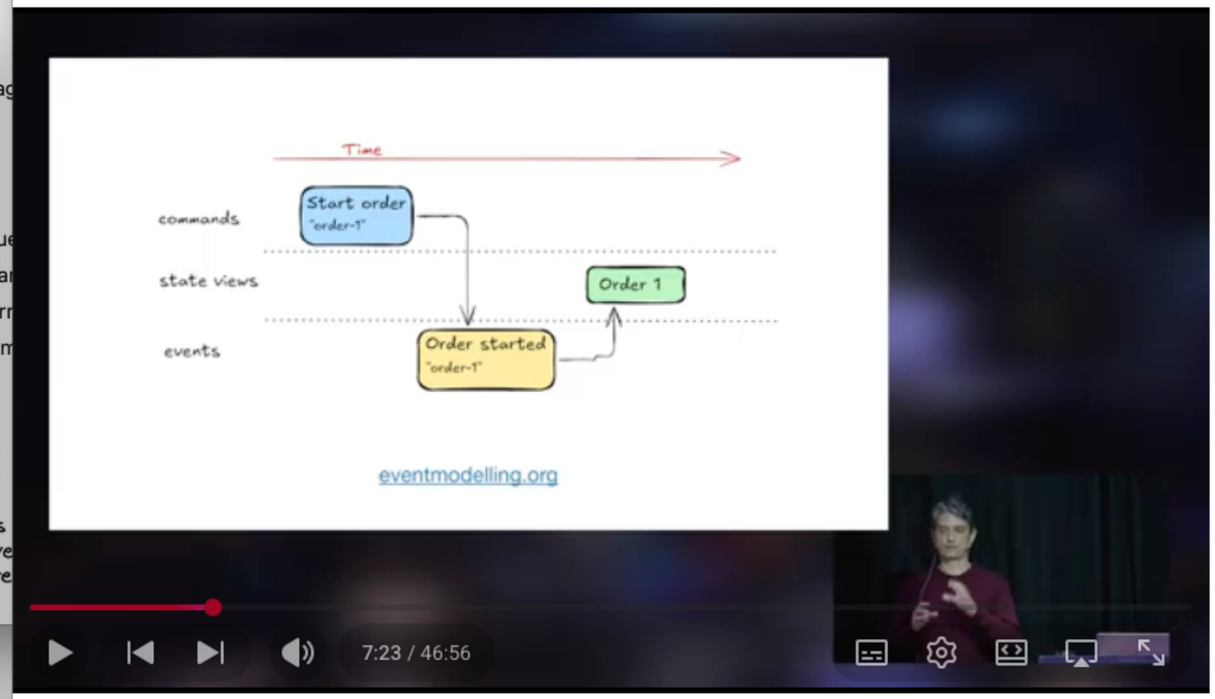
Then I use frameworks and programming paradigms that lean on this intuition. I model conceptual hierarchies as *associations*, and I think about domain entities as being *parents* or *children* of each other. I use statements like "a branch *has many* orders", "a line item *belongs to* an order".

www.youtube.com/watch?v=KKgc5T-qe5

YouTube GB

Search

Sign in



Next: Ismael Celis — Event-Sourced Mental Models in Ruby | Baltic Ruby 2025
Baltic Ruby 2025 - 19/22

Ismael Celis — Event-Sourced Mental Models in Ruby | Baltic Ruby 2025

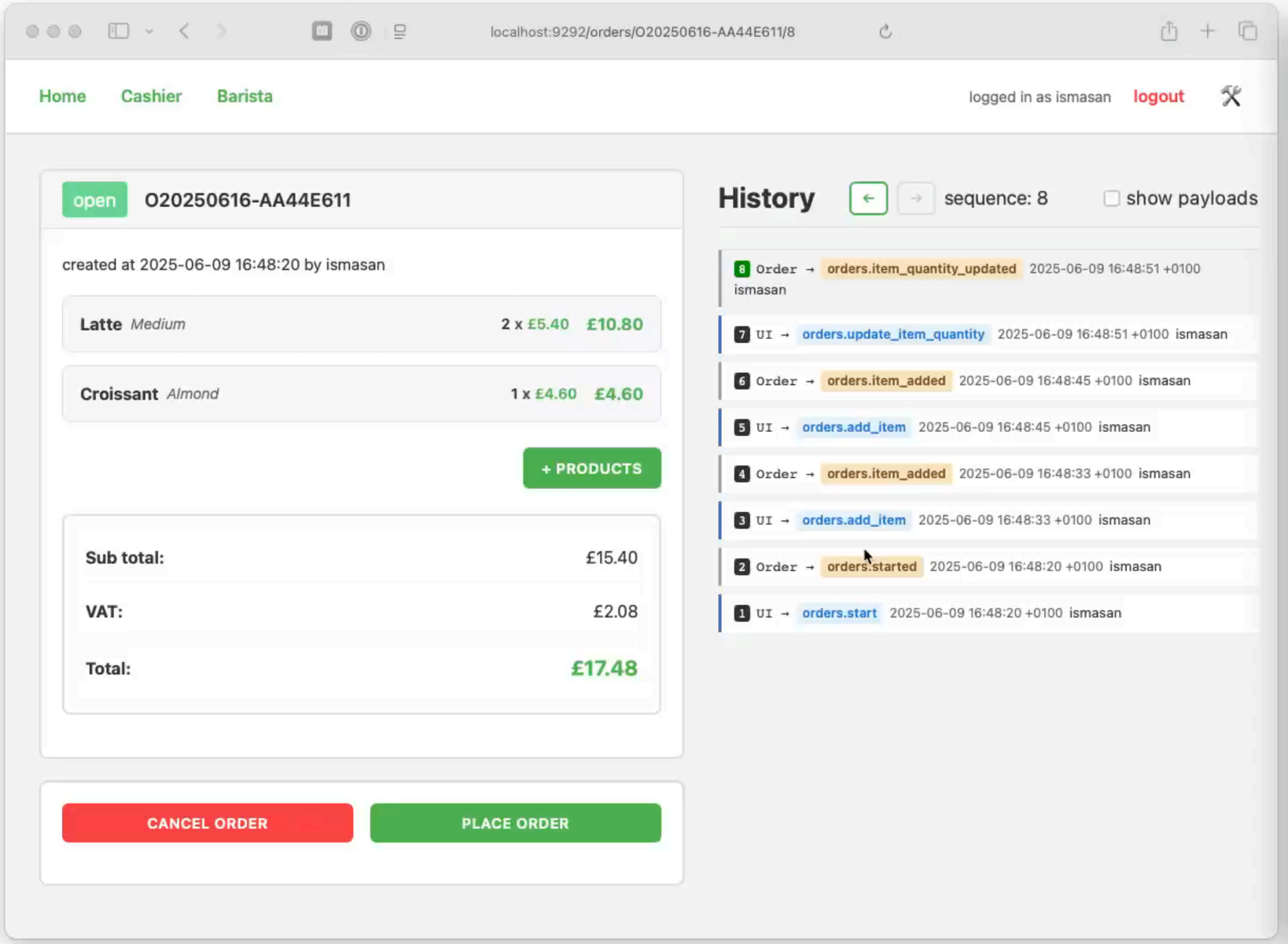
Unlisted

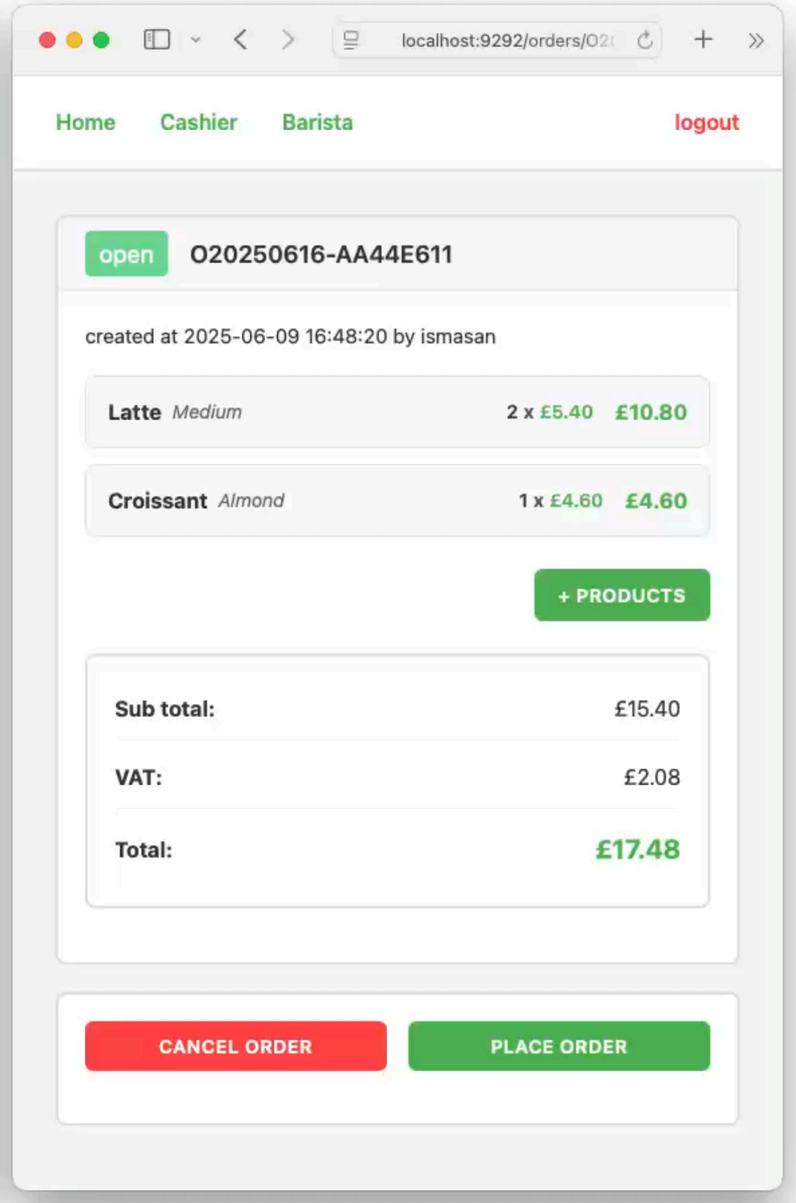
balticruby
205 subscribers

Subscribe

10 Share Save

347 views 2 months ago RIGA


localhost:9292/orders/O20250616-AA44E611/8
Home
Cashier
Barista
logged in as ismasan
logout
open
O20250616-AA44E611
created at 2025-06-09 16:48:20 by ismasan
Latte Medium
2 x £5.40
£10.80
Croissant Almond
1 x £4.60
£4.60
+ PRODUCTS
Sub total:
£15.40
VAT:
£2.08
Total:
£17.48
CANCEL ORDER
PLACE ORDER
History
sequence: 8
show payloads
8 Order → orders.item_quantity_updated 2025-06-09 16:48:51 +0100 ismasan
7 UI → orders.update_item_quantity 2025-06-09 16:48:51 +0100 ismasan
6 Order → orders.item_added 2025-06-09 16:48:45 +0100 ismasan
5 UI → orders.add_item 2025-06-09 16:48:45 +0100 ismasan
4 Order → orders.item_added 2025-06-09 16:48:33 +0100 ismasan
3 UI → orders.add_item 2025-06-09 16:48:33 +0100 ismasan
2 Order → orders.started 2025-06-09 16:48:20 +0100 ismasan
1 UI → orders.start 2025-06-09 16:48:20 +0100 ismasan


Home
Cashier
Barista
logout
open
O20250616-AA44E611
created at 2025-06-09 16:48:20 by ismasan
Latte Medium
2 x £5.40
£10.80
Croissant Almond
1 x £4.60
£4.60
+ PRODUCTS
Sub total:
£15.40
VAT:
£2.08
Total:
£17.48
CANCEL ORDER
PLACE ORDER

Events

```
document.addEventListener("click", function () {
  // do something
})
```

Not this.

Durable **Events**

```ruby
class OrderItem < ApplicationRecord
  after_save :log_event

  private

  def log_activity
    Event.create!(record: self, changes:)
  end
end
```

```ruby
# POST /orders/:order_id/items
def create
  OrderItem.transaction do
    item = order.order_items.create!(item_params)
    Event.log!(item, ‘orders.product_added’)
    OrderItemProcessingJob.perform_later(item)
  end
end
```

```ruby
ProductAdded = Data.define(:product_id, :quantity)
DiscountAdded = Data.define(:product_id, :amount, :reference)


# POST /orders/:order_id/items
def create
  event = ProductAdded.new(**item_params)

  item = order.process_event(event)

  OrderItemProcesingJob.perform_later(item)
end
```

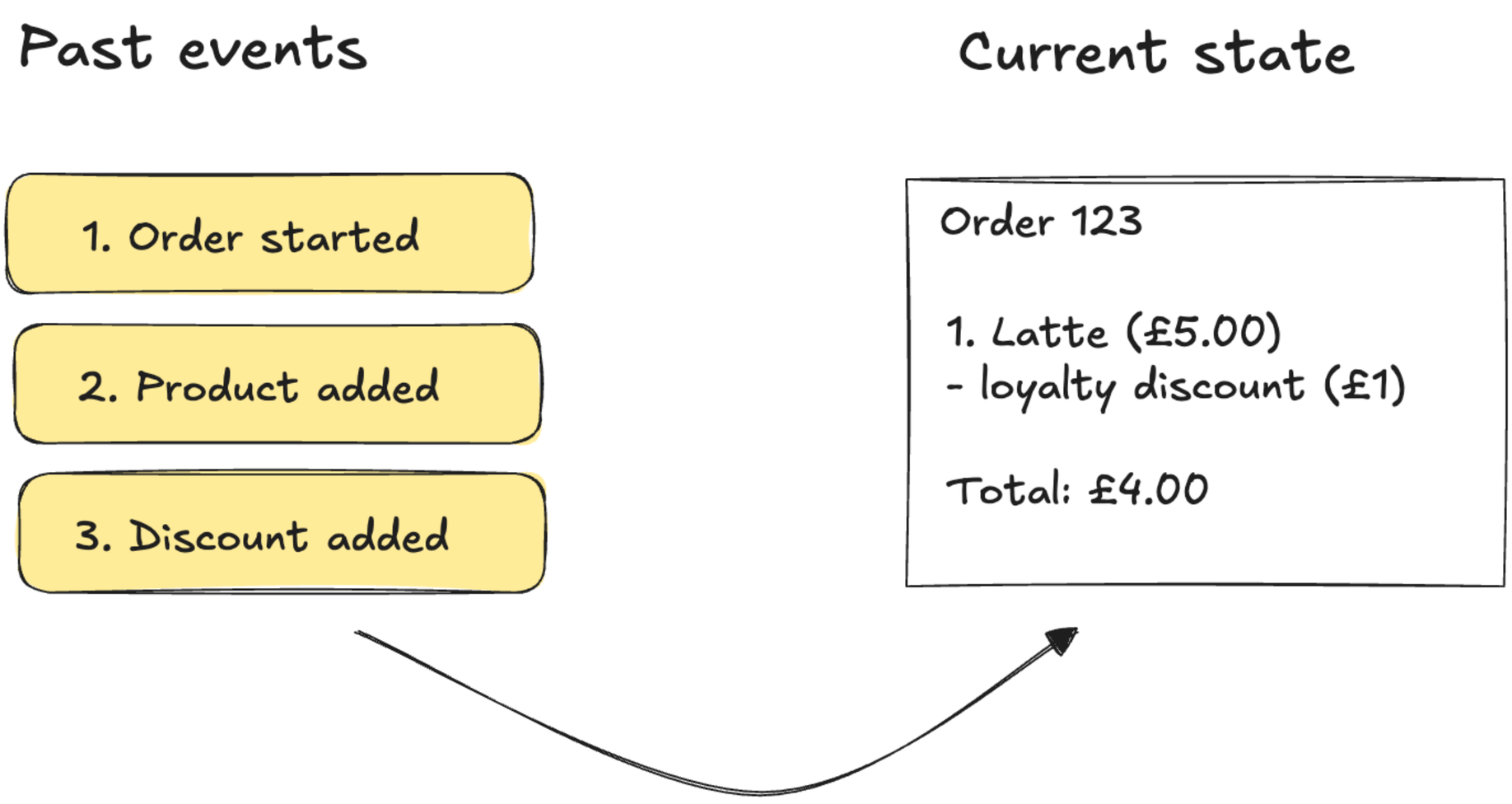
```ruby
class Order < ApplicationRecord

  def process_event(event)
    Order.transaction do
      case event

      when ProductAdded
        order_items.create!(product_id: event.product_id, amount: event.amount,…)

      when DiscountAdded
        item = order_items.find_by(product_id: event.product_id)
        item.add_discount(event.amount, event.reference)
      end

      Event.create!(payload: event)
    end
  end

end
```

PaymentConfirmed

Durable Domain Events

DiscountAdded

What’s the point of state?

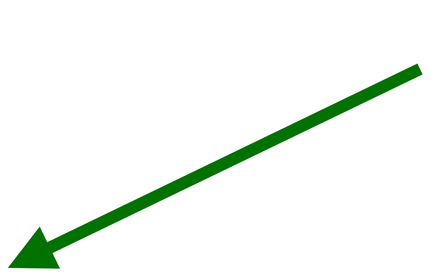

Past events
1. Order started
2. Product added
3. Discount added
Current state
Order 123
1. Latte (£5.00)
- loyalty discount (£1)
Total: £4.00

```
OrderStarted = Data.define()
ProductAdded = Data.define(:product_id, :quantity, :price)
DiscountAdded = Data.define(:product_id, :amount, :reference)
```

```
events = [
  OrderStarted.new(),
  ProductAdded.new('latte', 1, 500),
  DiscountAdded.new('latte', 100, 'loyalty-program')
]

EventStore.append('order-123', events)
```

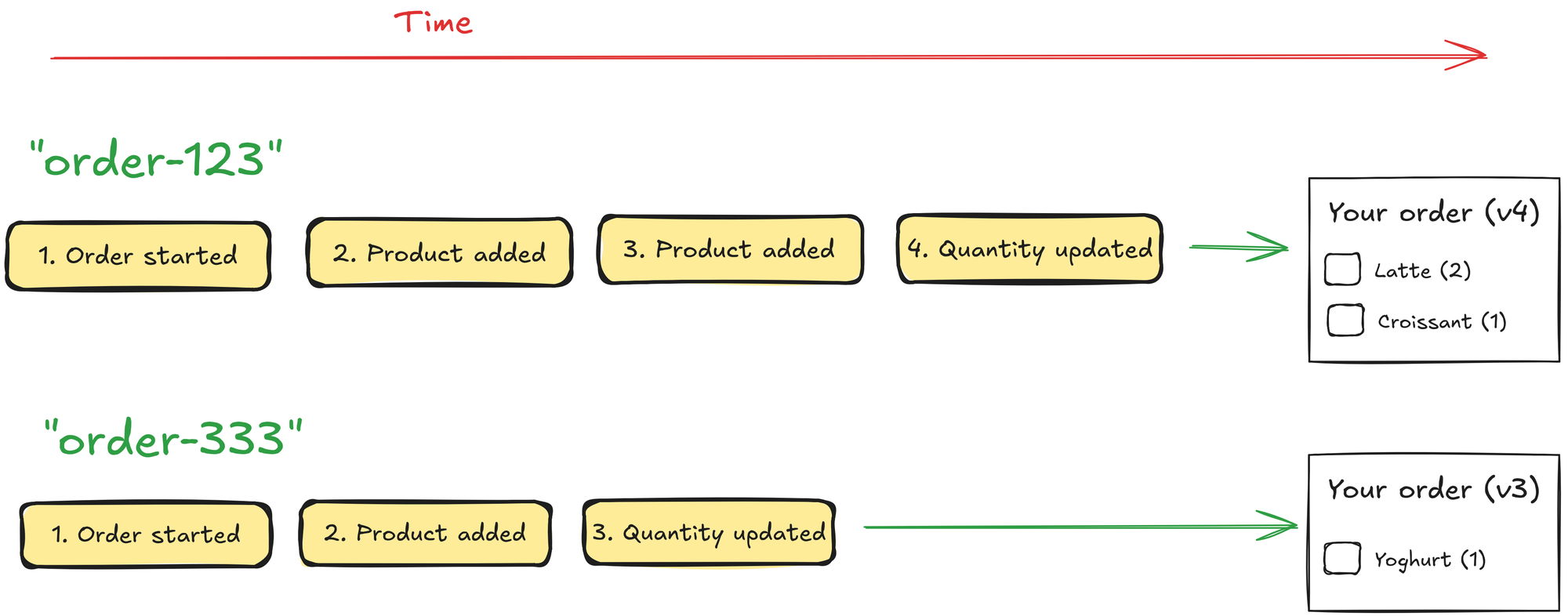
```ruby
events = EventStore.read('order-123')
order = Order.new # <= this is a PORO


def evolve(order, events)
  events.each do |event|
    case event
    when OrderStarted
      order.status = 'started'

    when ProductAdded
      order.add_product(event.product_id, event.quantity, …)

    when DiscountAdded
      order.add_product_discount(event.product_id, event.amount, …)
    end
  end
end
```

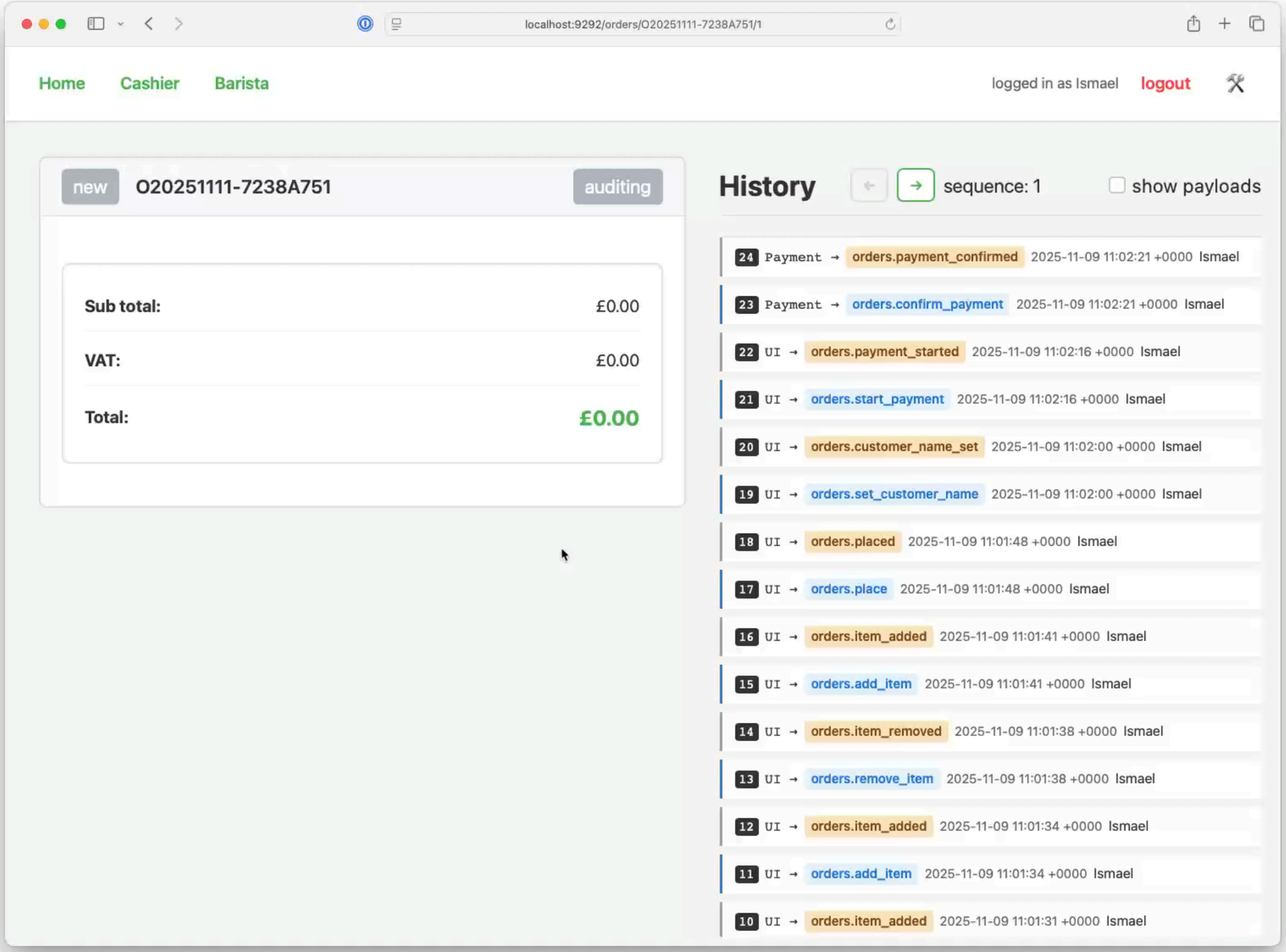
Initial	€0.00
Travel budget	+ €1,000.00
Flight	- €300.00
Hotel room	- €100.00
Bus ticket	- €1.50
Balance	€598.50



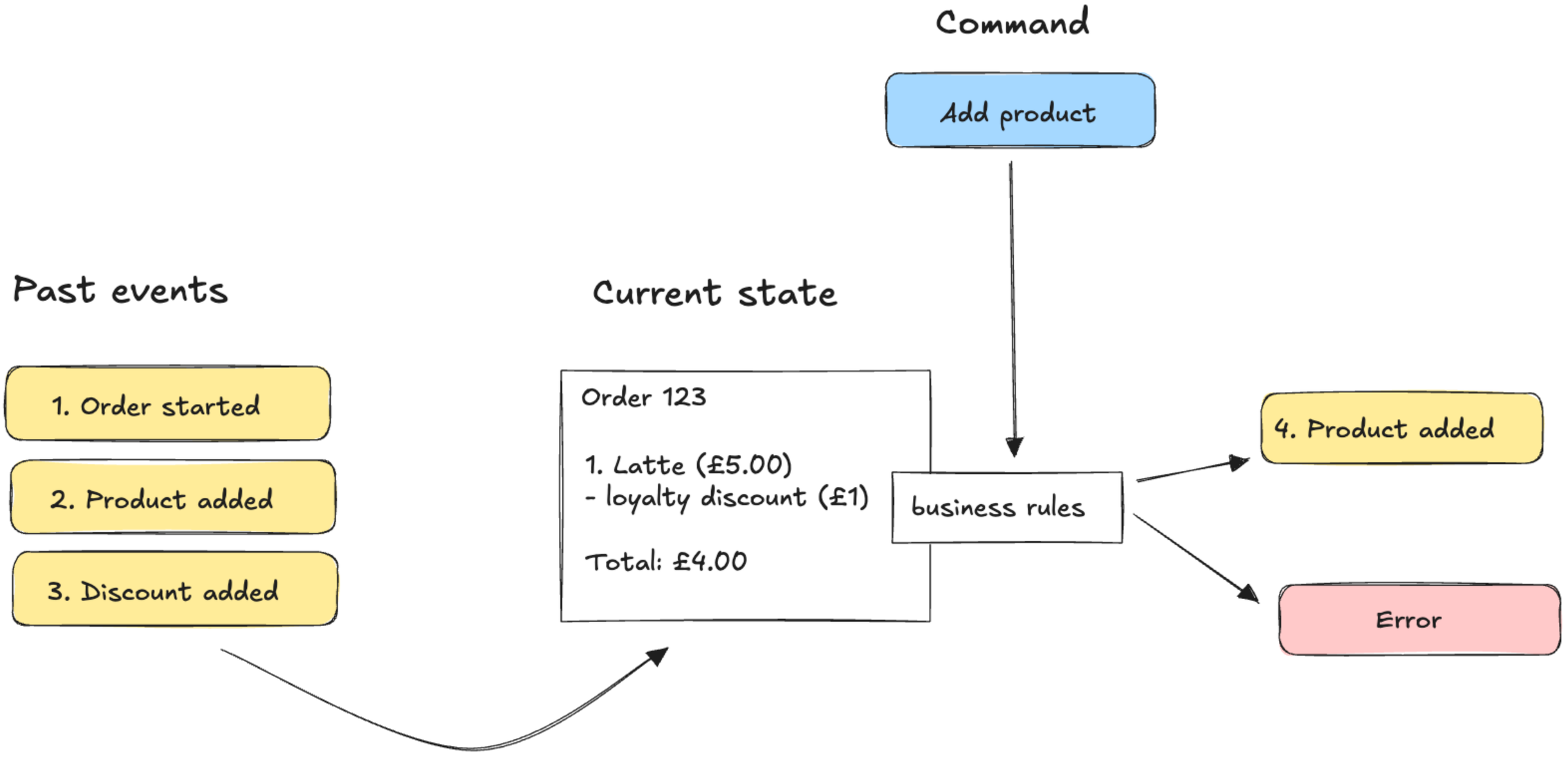
Events

Current state


Time
"order-123"
1. Order started
2. Product added
3. Product added
4. Quantity updated
Your order (v4)
Latte (2)
Croissant (1)
"order-333"
1. Order started
2. Product added
3. Quantity updated
Your order (v3)
Yoghurt (1)


localhost:9292/orders/O20251111-7238A751/1
Home
Cashier
Barista
logged in as Ismael
logout
new
O20251111-7238A751
auditing
Sub total:
£0.00
VAT:
£0.00
Total:
£0.00
History
sequence: 1
show payloads
24 Payment → orders.payment_confirmed 2025-11-09 11:02:21 +0000 Ismael
23 Payment → orders.confirm_payment 2025-11-09 11:02:21 +0000 Ismael
22 UI → orders.payment_started 2025-11-09 11:02:16 +0000 Ismael
21 UI → orders.start_payment 2025-11-09 11:02:16 +0000 Ismael
20 UI → orders.customer_name_set 2025-11-09 11:02:00 +0000 Ismael
19 UI → orders.set_customer_name 2025-11-09 11:02:00 +0000 Ismael
18 UI → orders.placed 2025-11-09 11:01:48 +0000 Ismael
17 UI → orders.place 2025-11-09 11:01:48 +0000 Ismael
16 UI → orders.item_added 2025-11-09 11:01:41 +0000 Ismael
15 UI → orders.add_item 2025-11-09 11:01:41 +0000 Ismael
14 UI → orders.item_removed 2025-11-09 11:01:38 +0000 Ismael
13 UI → orders.remove_item 2025-11-09 11:01:38 +0000 Ismael
12 UI → orders.item_added 2025-11-09 11:01:34 +0000 Ismael
11 UI → orders.add_item 2025-11-09 11:01:34 +0000 Ismael
10 UI → orders.item_added 2025-11-09 11:01:31 +0000 Ismael


Add product
[business rules]
Product added

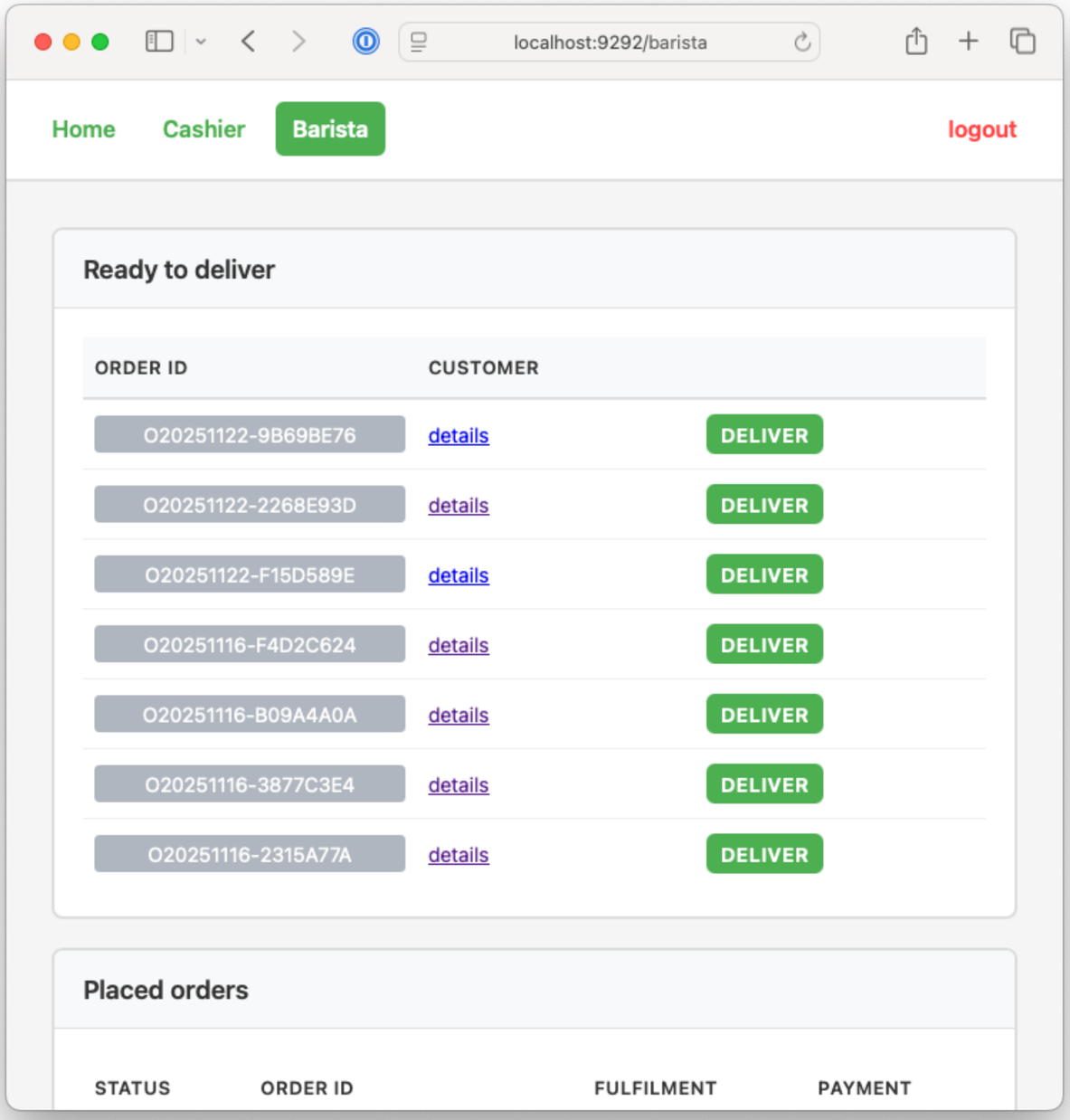

Command
Add product
Past events
1. Order started
2. Product added
3. Discount added
Current state
Order 123
1. Latte (£5.00)
- loyalty discount (£1)
Total: £4.00
business rules
4. Product added
Error

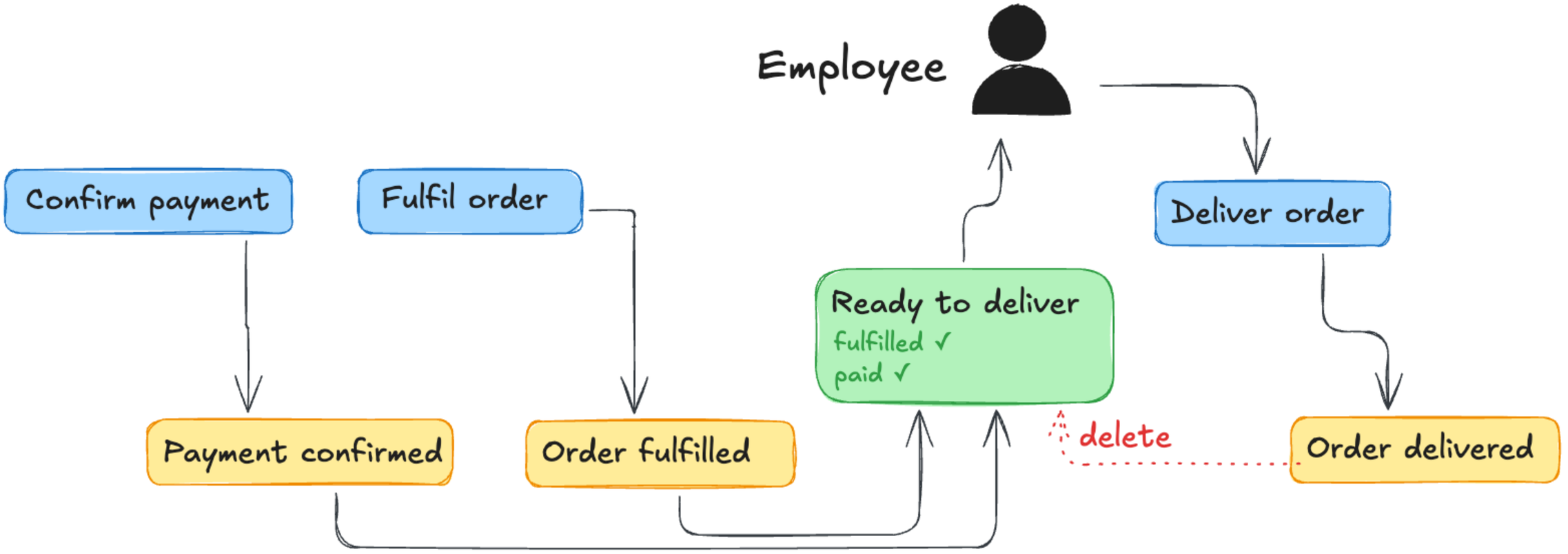
```ruby
AddProduct = Data.define(:order_id, :product_id, :quantity, :price)
ProductAdded = Data.define(:product_id, :quantity, :price)

command = AddProduct.new('order-123', 'latte', 1, 500)

order = Order.new

past_events = EventStore.read(command.order_id)

evolve(order, events) # <= build up state from past events

new_events = decide(order, command) # <= produce new events

EventStore.append(command.order_id, new_events)
```

```ruby
AddProduct = Data.define(:order_id, :product_id, :quantity, :price)
ProductAdded = Data.define(:product_id, :quantity, :price)

command = AddProduct.new('order-123', 'latte', 1, 500)

order = Order.new

past_events = EventStore.read(command.order_id)

evolve(order, events) # <= build up state from past events

new_events = decide(order, command) # <= produce new events

EventStore.append(command.order_id, new_events)
```

```
AddProduct = Data.define(:order_id, :product_id, :quantity, :price)
ProductAdded = Data.define(:product_id, :quantity, :price)

command = AddProduct.new('order-123', 'latte', 1, 500)

order = Order.new

past_events = EventStore.read(command.order_id)

evolve(order, events) # <= build up state from past events

new_events = decide(order, command) # <= produce new events

EventStore.append(command.order_id, new_events)
```

```ruby
def decide(order, command)
  new_events = []

  case command
  when AddProduct
    if order.open?
      new_events << ProductAdded.new(command.product_id, ...)
    end
  when RemoveProduct
    # etc
  end

  new_events
end
```

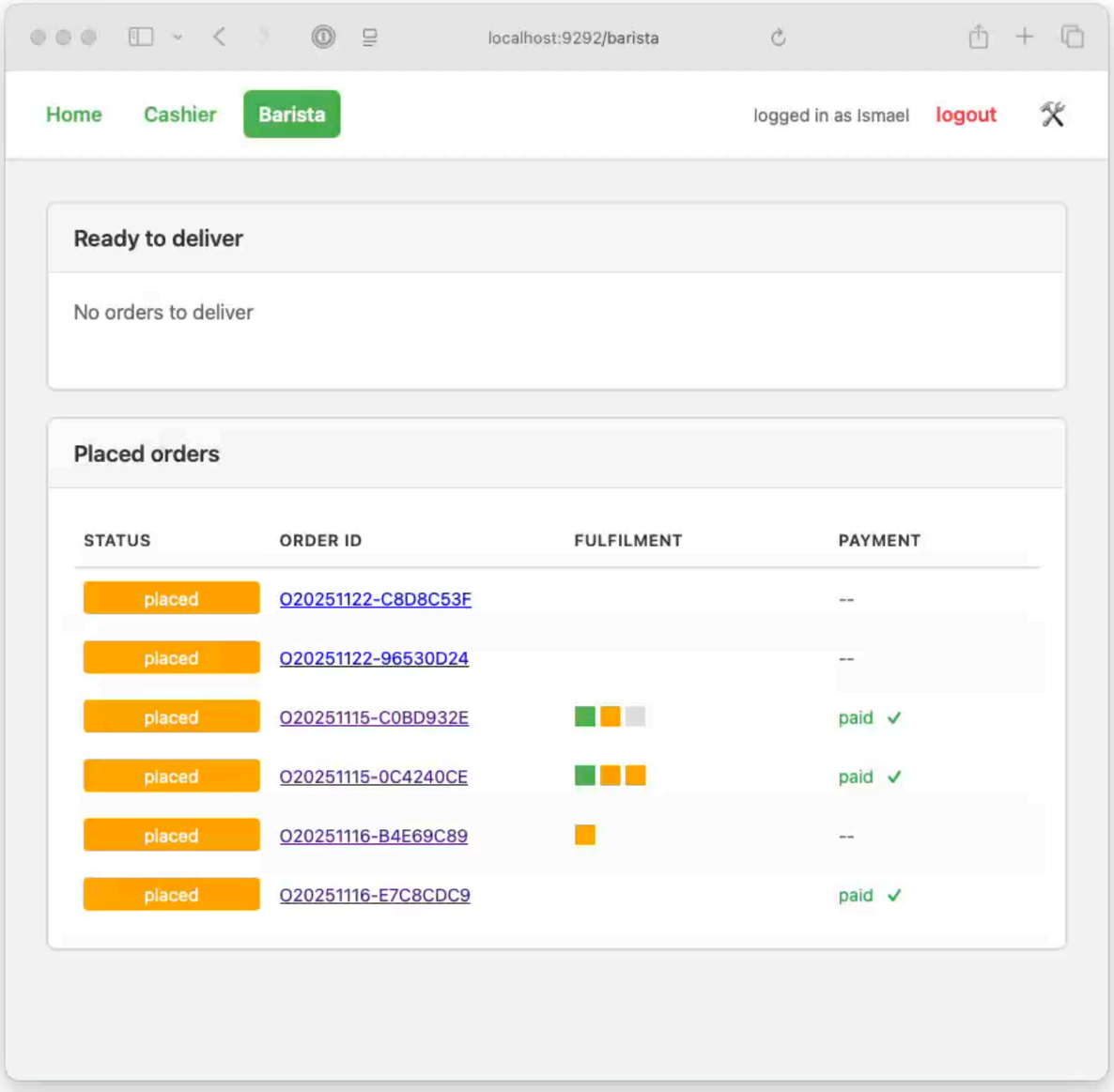

localhost:9292/barista
Home
Cashier
Barista
logout
Ready to deliver
ORDER ID
CUSTOMER
O20251122-9B69BE76
details
DELIVER
O20251122-2268E93D
details
DELIVER
O20251122-F15D589E
details
DELIVER
O20251116-F4D2C624
details
DELIVER
O20251116-B09A4A0A
details
DELIVER
O20251116-3877C3E4
details
DELIVER
O20251116-2315A77A
details
DELIVER
Placed orders
STATUS
ORDER ID
FULFILMENT
PAYMENT

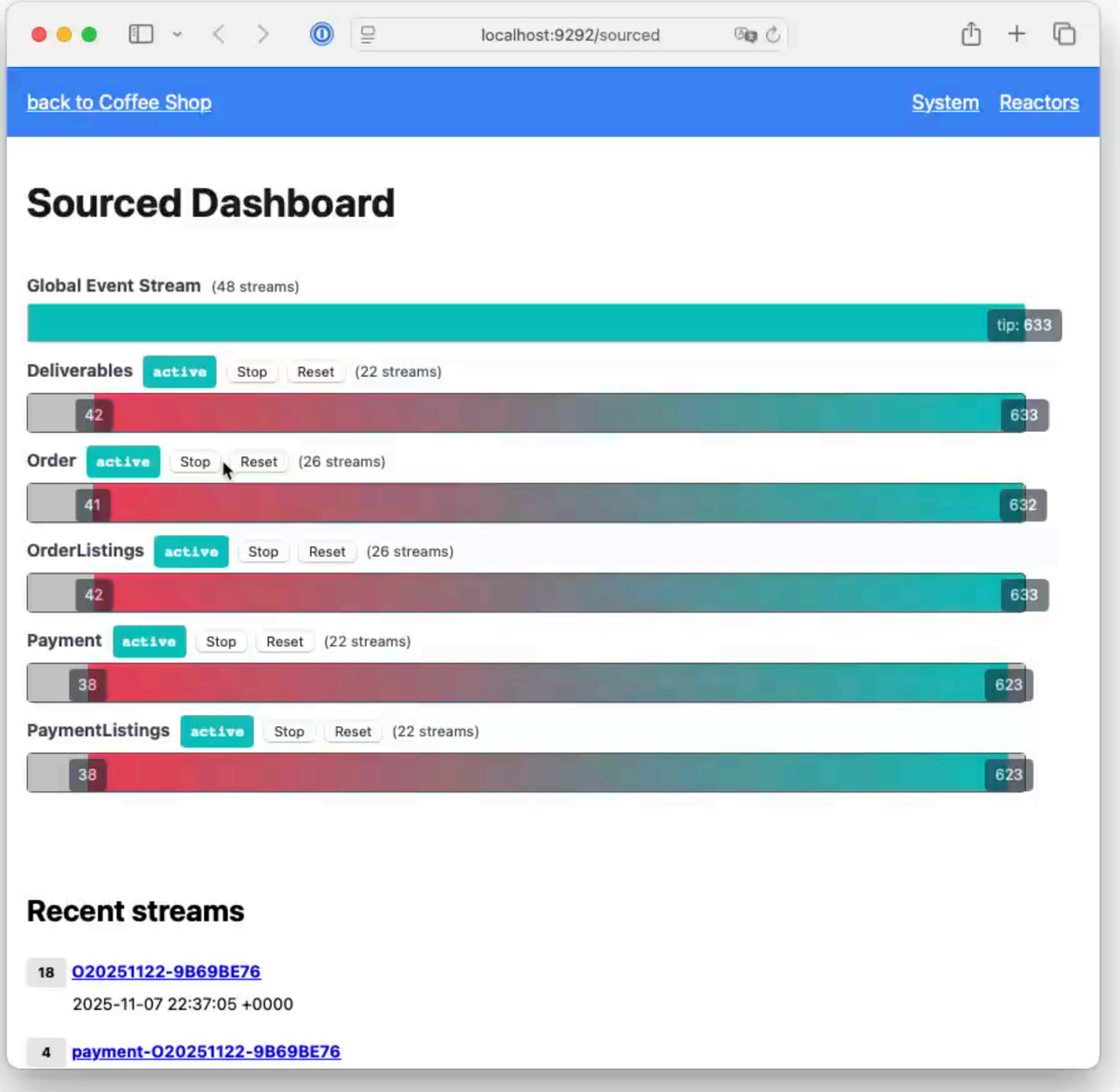

Employee
Confirm payment
Fulfil order
Deliver order
Ready to deliver
fulfilled ✓
paid ✓
Payment confirmed
Order fulfilled
delete
Order delivered

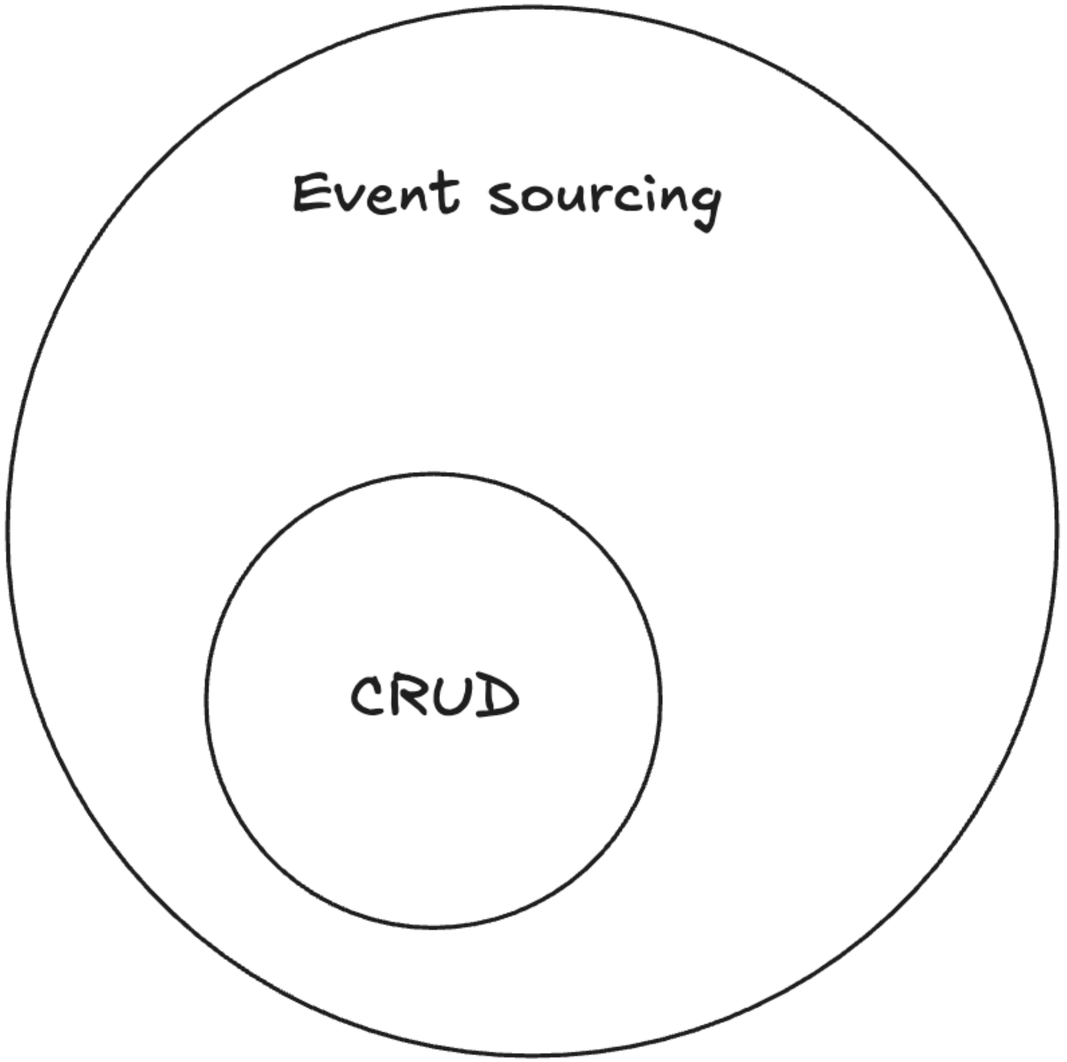
```ruby
event Order::OrderFulfilled do |state, event|
  state[:fulfilled] = true
  check_deliverable(state)
end

event Order::PaymentConfirmed do |state, event|
  state[:paid] = true
  check_deliverable(state)
end

def check_deliverable(state)
  state[:status] = 'ready' if state[:fulfilled] && state[:paid]
end
```

```ruby
# This block runs in a transaction when handling events

sync do |state:, events:, replaying:|
  path = File.join(DATA_DIR, "#{state[:id]}.json")

  File.write(path, JSON.pretty_generate(state.to_h))
end
```

```ruby
event Order::CustomerNameSet do |state, event|
  state[:customer_name] = event.payload.customer_name
end
```


localhost:9292/barista
Home
Cashier
Barista
logged in as Ismael
logout
Ready to deliver
No orders to deliver
Placed orders
STATUS
ORDER ID
FULFILMENT
PAYMENT
placed
O20251122-C8D8C53F
--
placed
O20251122-96530D24
--
placed
O20251115-C0BD932E
paid ✓
placed
O20251115-0C4240CE
paid ✓
placed
O20251116-B4E69C89
--
placed
O20251116-E7C8CDC9
paid ✓


localhost:9292/sourced
back to Coffee Shop
System
Reactors
Sourced Dashboard
Global Event Stream (48 streams)
tip: 633
Deliverables active Stop Reset (22 streams)
42
633
Order active Stop Reset (26 streams)
41
632
OrderListings active Stop Reset (26 streams)
42
633
Payment active Stop Reset (22 streams)
38
623
PaymentListings active Stop Reset (22 streams)
38
623
Recent streams
18 O20251122-9B69BE76
2025-11-07 22:37:05 +0000
4 payment-O20251122-9B69BE76


Event sourcing
CRUD

localhost:9292/barista

Barista - Sourced Coffee

Home Cashier **Barista** logout

Placed orders

STATUS	ORDER ID	FULFILMENT	PAYMENT
placed	O20251116-C82D028A		--
placed	O20251115-C0BD932E		paid ✓
placed	O20251115-0C4240CE		paid ✓
placed	O20251116-B4E69C89		--
placed	O20251116-E7C8CDC9		--

Ready to deliver

No orders to deliver

localhost:9292/orders/O20251116-C82D028A

new O20251116-C82D028A - Sourced Coffee

placed **O20251116-C82D028A**

created at 2025-11-06 16:09:01 by Ismael fulfillment

Espresso *Double Shot* 1 x £3.60 **£3.60**

Cappuccino *Small* 1 x £4.20 **£4.20**

Sub total:	£7.80
VAT:	£1.05
Total:	**£8.85**

pending **Payment**

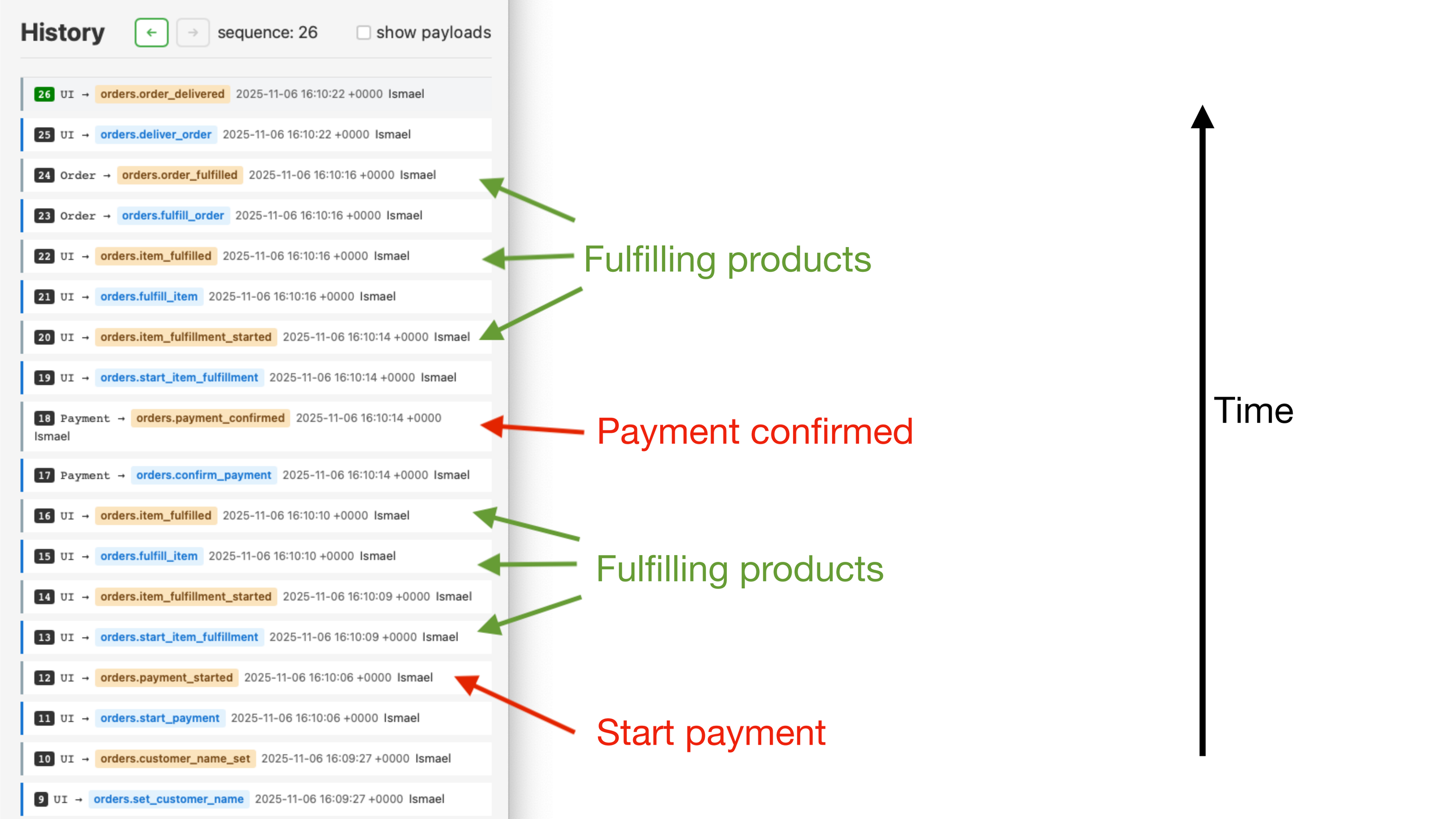
Customer name EDIT

Joe

History

sequence: 10 show payloads

10 UI → orders.customer_name_set 2025-11-06 16:09:27 +0000 Ismael
9 UI → orders.set_customer_name 2025-11-06 16:09:27 +0000 Ismael
8 UI → orders.placed 2025-11-06 16:09:17 +0000 Ismael
7 UI → orders.place 2025-11-06 16:09:17 +0000 Ismael
6 UI → orders.item_added 2025-11-06 16:09:10 +0000 Ismael
5 UI → orders.add_item 2025-11-06 16:09:10 +0000 Ismael
4 UI → orders.item_added 2025-11-06 16:09:07 +0000 Ismael
3 UI → orders.add_item 2025-11-06 16:09:07 +0000 Ismael
2 UI → orders.started 2025-11-06 16:09:01 +0000 Ismael
1 UI → orders.start 2025-11-06 16:09:01 +0000 Ismael


History
sequence: 26
show payloads
26 UI → orders.order_delivered 2025-11-06 16:10:22 +0000 Ismael
25 UI → orders.deliver_order 2025-11-06 16:10:22 +0000 Ismael
24 Order → orders.order_fulfilled 2025-11-06 16:10:16 +0000 Ismael
23 Order → orders.fulfill_order 2025-11-06 16:10:16 +0000 Ismael
22 UI → orders.item_fulfilled 2025-11-06 16:10:16 +0000 Ismael
21 UI → orders.fulfill_item 2025-11-06 16:10:16 +0000 Ismael
20 UI → orders.item_fulfillment_started 2025-11-06 16:10:14 +0000 Ismael
19 UI → orders.start_item_fulfillment 2025-11-06 16:10:14 +0000 Ismael
18 Payment → orders.payment_confirmed 2025-11-06 16:10:14 +0000 Ismael
17 Payment → orders.confirm_payment 2025-11-06 16:10:14 +0000 Ismael
16 UI → orders.item_fulfilled 2025-11-06 16:10:10 +0000 Ismael
15 UI → orders.fulfill_item 2025-11-06 16:10:10 +0000 Ismael
14 UI → orders.item_fulfillment_started 2025-11-06 16:10:09 +0000 Ismael
13 UI → orders.start_item_fulfillment 2025-11-06 16:10:09 +0000 Ismael
12 UI → orders.payment_started 2025-11-06 16:10:06 +0000 Ismael
11 UI → orders.start_payment 2025-11-06 16:10:06 +0000 Ismael
10 UI → orders.customer_name_set 2025-11-06 16:09:27 +0000 Ismael
9 UI → orders.set_customer_name 2025-11-06 16:09:27 +0000 Ismael
Fulfilling products
Payment confirmed
Fulfilling products
Start payment
Time

Eventual Consistency

MING VASE

RUBRUM

ORIENTE ITALIANO

£700

Dimensions

↕ 9.84 inch / ⌀ max. 6.30 inch / 2.65 pound

AVAILABLE IN 11 OTHER DESIGNS

1 + ADD TO CART

Exclusive box included | Made in Italy | Pure gold

Description

Ming porcelain vase with red and pure gold Rubrum decoration. Oriente Italiano Gold Collection.

Expression of the savoir faire of excellence and Italian craftsmanship, the Ming vase combines the mastery of the Ginori 1735 artisans with oriental lines, for a collector's art object, in which the decoration enhances the intense red of the porcelain in contrast with the floral motif in pure gold, making it the protagonist in every environment. Also available in the refined Aurum variant in white and gold, it completes the proposal of the Oriente Italiano Gold Collection, in an assortment of tableware, tea and coffee, other furnishing accessories and sophisticated room

This item is shipped in 1-3 working days.
Discover our **delivery times.**

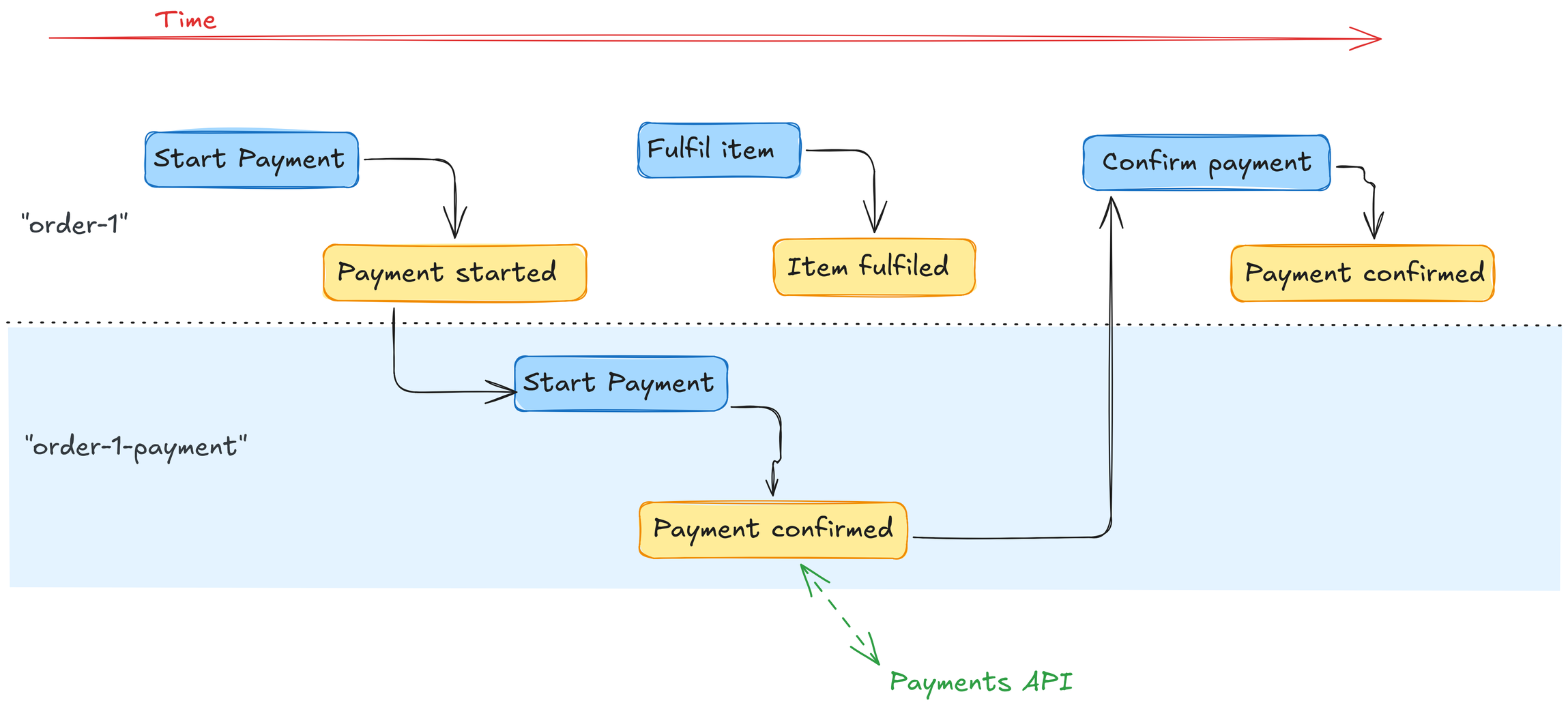
Maintenance

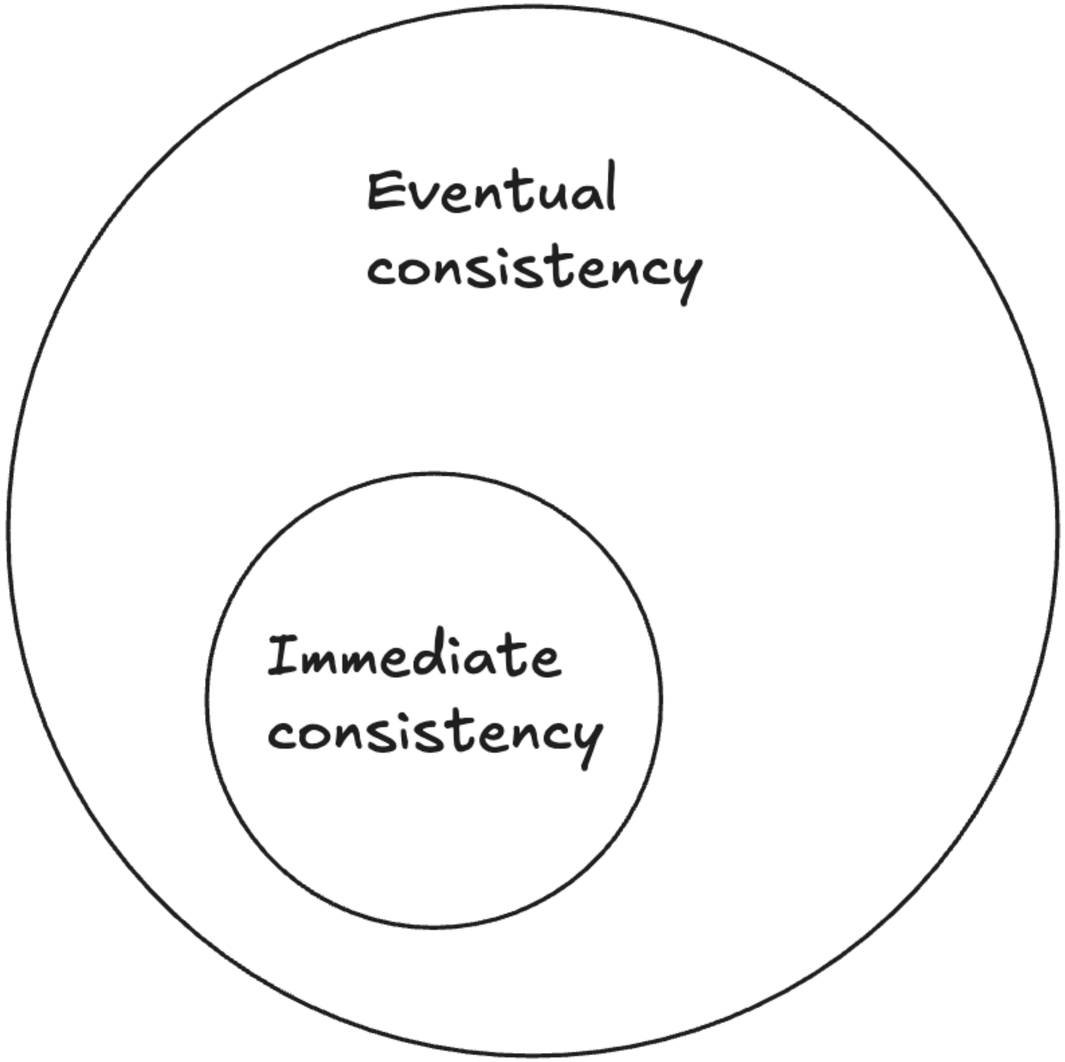
Do not microwave

Hand-wash only

```ruby
# POST /orders/:order_id/payment
def create
  order = Order.find(params[:order_id])
  payment = Payment.start(params[:params])
  PaymentProcessingJob.perform_later(payment)
  redirect_to order_url(order)
end
```

Just model the workflow!




Eventual
consistency
Immediate
consistency

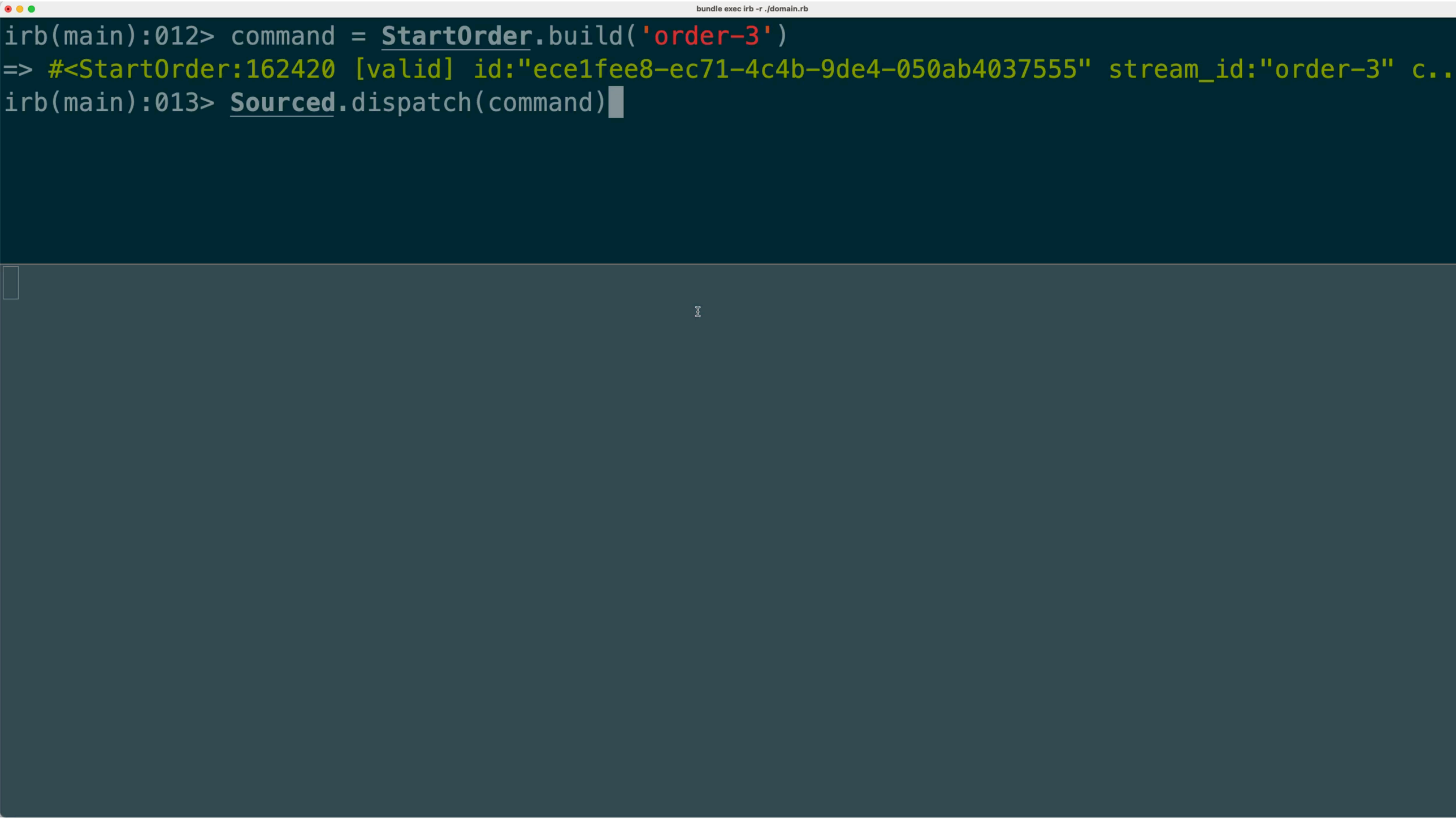
The runtime

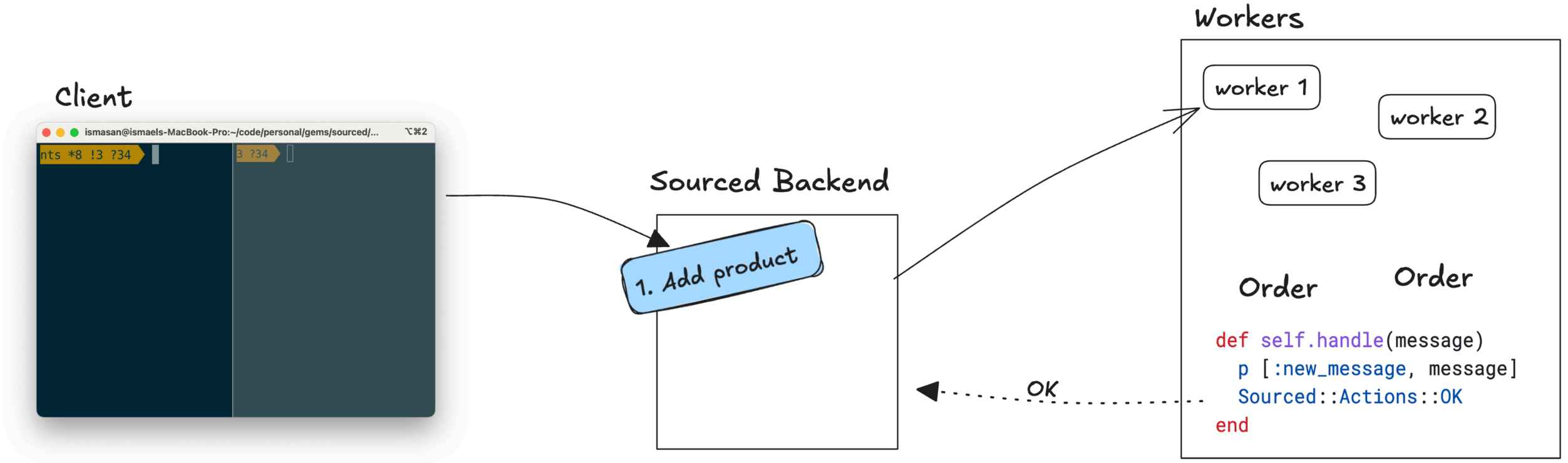
github.com/ismasan/sourced

```ruby
# ./domain.rb
require 'sourced'

class Order
  extend Sourced::Consumer

  def self.handled_messages
    [Start, AddProduct, RemoveProduct]
  end

  def self.handle(message)
    p [:new_message, message.inspect]
    Sourced::Actions::OK
  end
end
```

```
# bundle exec ruby ./bin/workers.rb


require_relative './../domain'

Sourced::Supervisor.start(count: 10)
```

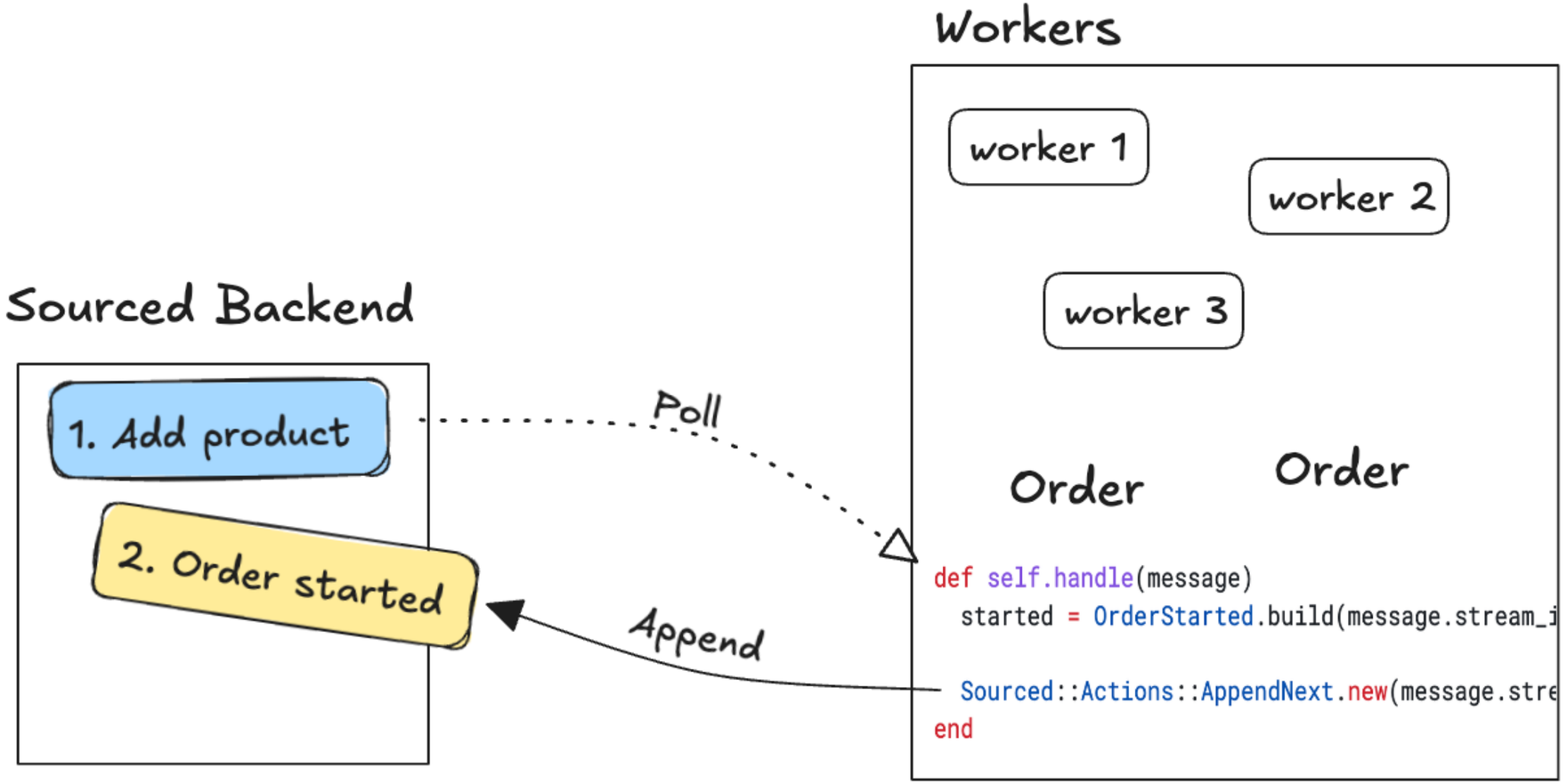
```
# bundle exec irb -r ./domain.rb

command = StartOrder.build('order-123')

Sourced.dispatch(command)
```

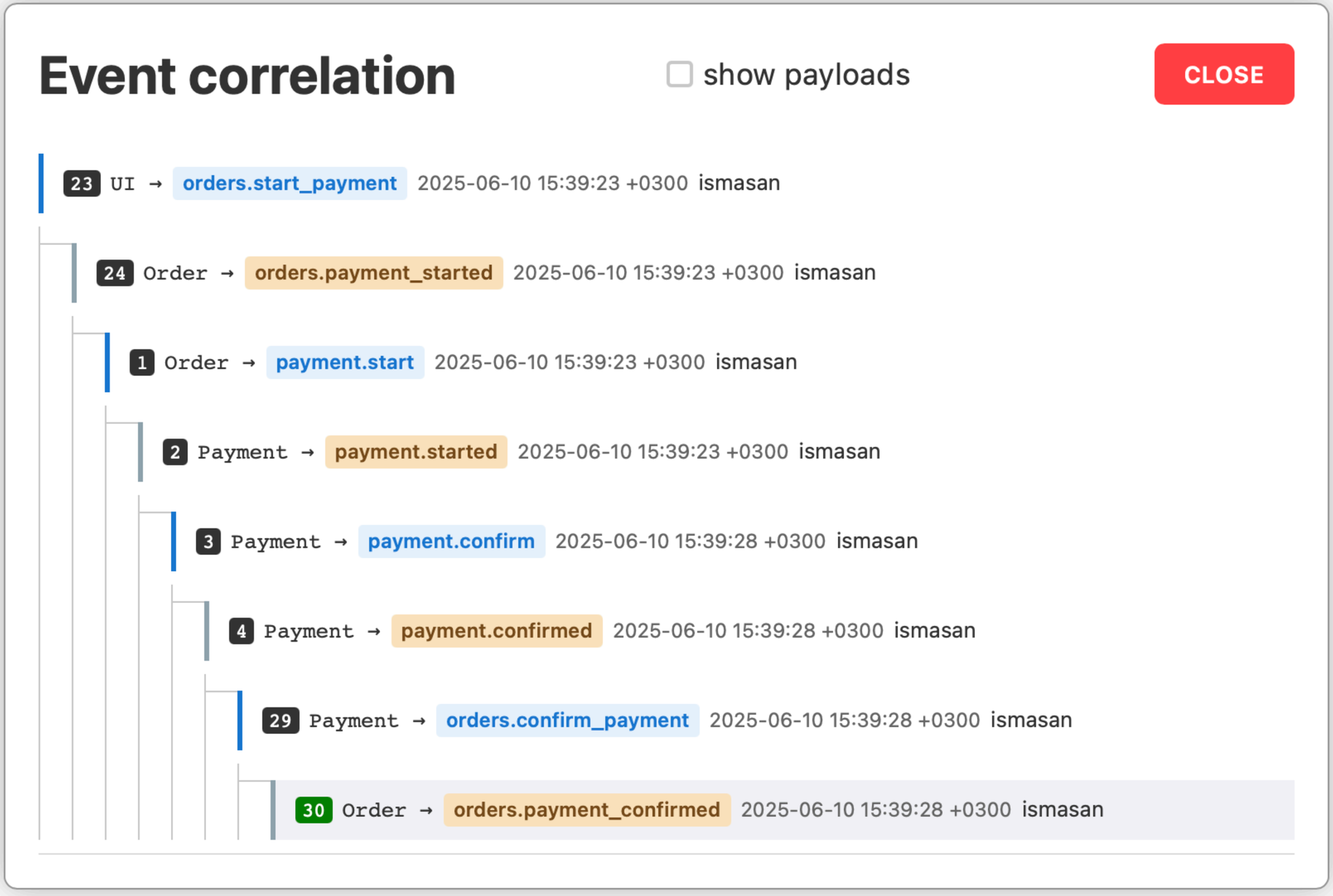

bundle exec irb -r ./domain.rb
irb(main):012> command = StartOrder.build('order-3')
=> #<StartOrder:162420 [valid] id:"ece1fee8-ec71-4c4b-9de4-050ab4037555" stream_id:"order-3" c..
irb(main):013> Sourced.dispatch(command)


Client
ismasan@ismaels-MacBook-Pro:~/code/personal/gems/sourced/...
nts *8 !3 ?34
3 ?34
Sourced Backend
1. Add product
OK
Workers
worker 1
worker 2
worker 3
Order
Order
def self.handle(message)
p [:new_message, message]
Sourced::Actions::OK
end

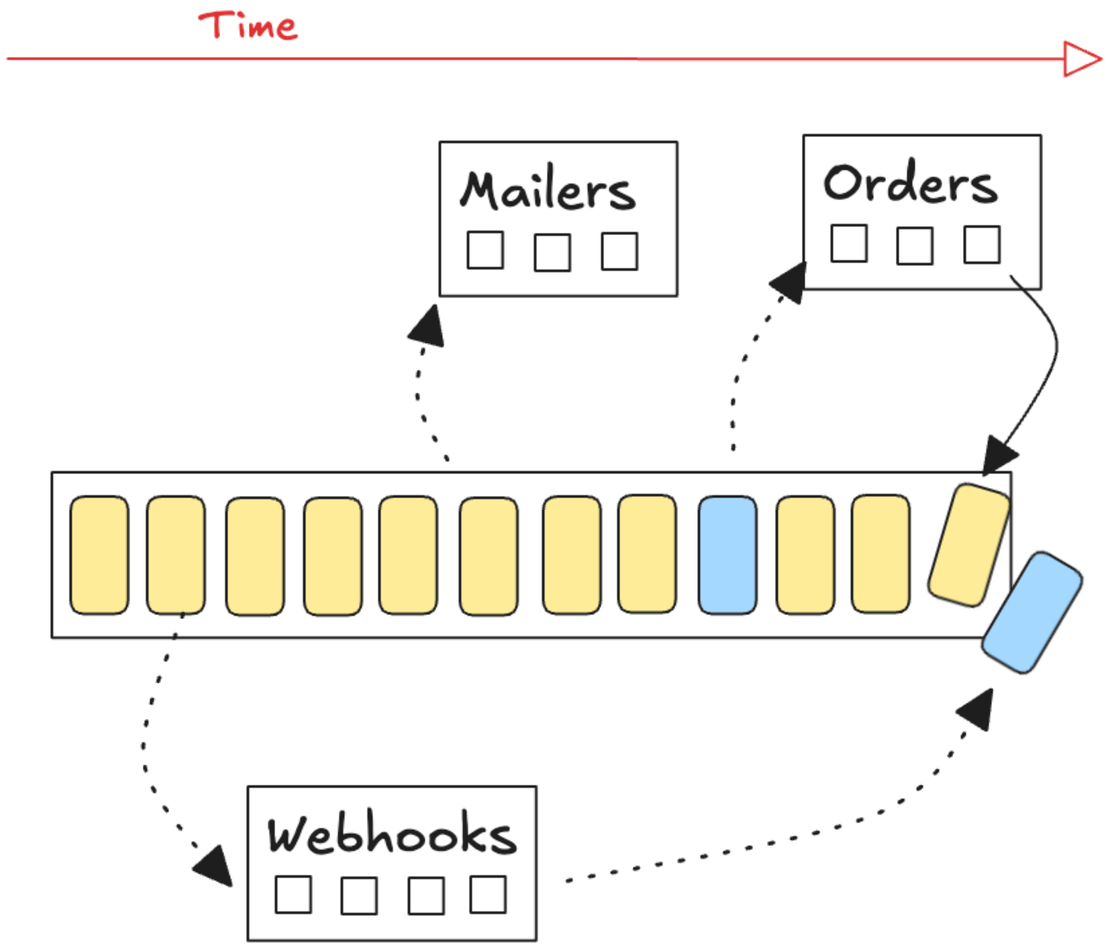
```ruby
def self.handle(message)
  started = OrderStarted.build(message.stream_id)

  Sourced::Actions::AppendNext.new([started])
end
```


ems/sourced/...
⌥⌘2


Workers
worker 1
worker 2
worker 3
Order
Order
Sourced Backend
1. Add product
2. Order started
Poll
Append
def self.handle(message)
started = OrderStarted.build(message.stream_i
Sourced::Actions::AppendNext.new(message.stre
end


Event correlation
show payloads
CLOSE
23 UI → orders.start_payment 2025-06-10 15:39:23 +0300 ismasan
24 Order → orders.payment_started 2025-06-10 15:39:23 +0300 ismasan
1 Order → payment.start 2025-06-10 15:39:23 +0300 ismasan
2 Payment → payment.started 2025-06-10 15:39:23 +0300 ismasan
3 Payment → payment.confirm 2025-06-10 15:39:28 +0300 ismasan
4 Payment → payment.confirmed 2025-06-10 15:39:28 +0300 ismasan
29 Payment → orders.confirm_payment 2025-06-10 15:39:28 +0300 ismasan
30 Order → orders.payment_confirmed 2025-06-10 15:39:28 +0300 ismasan

Autonomy



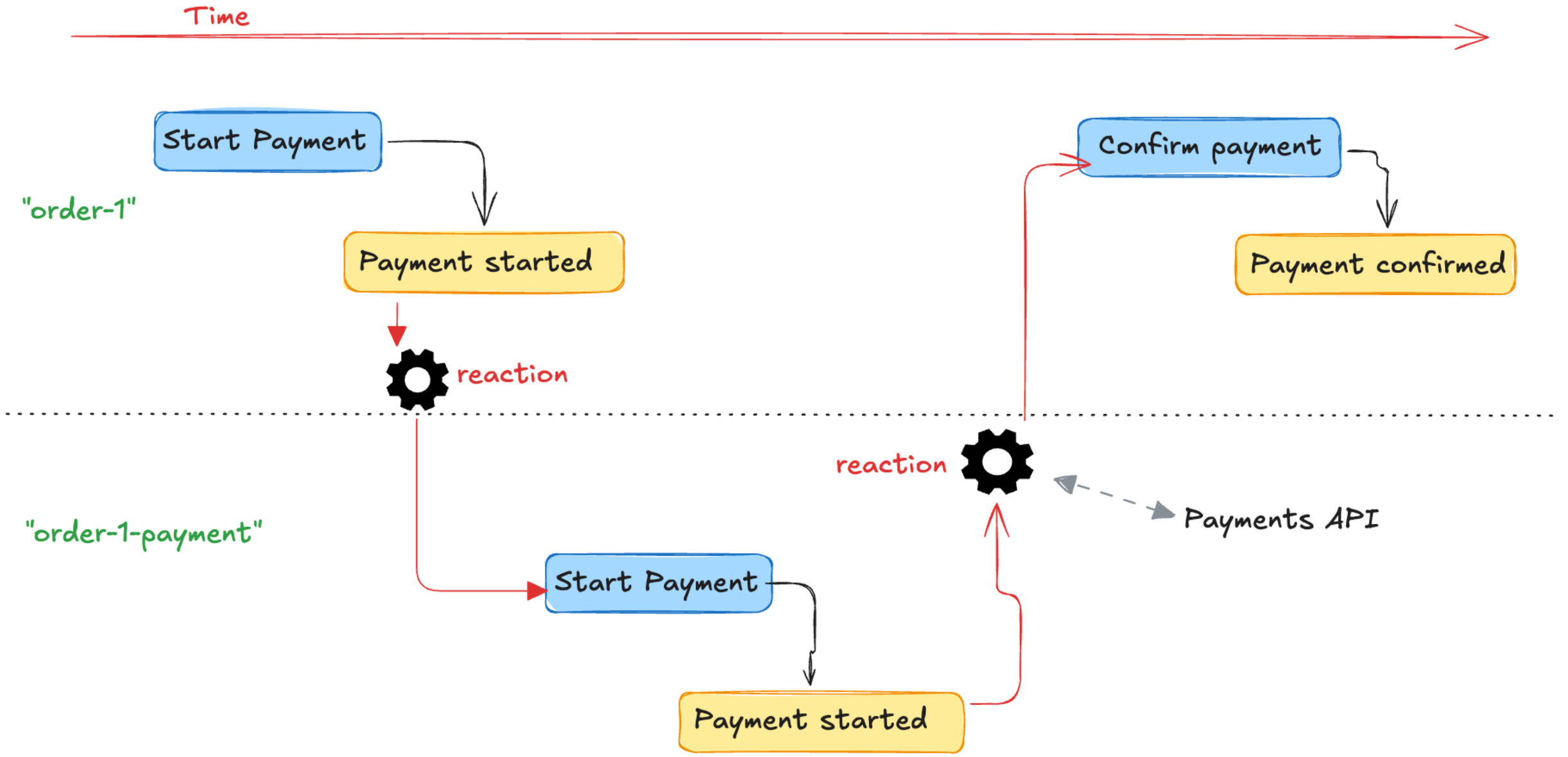
```ruby
class Webhooks
  extend Sourced::Consumer

  def self.handled_messages
    [OrderStarted, OrderPlaced]
  end

  def self.handle(message)
    HTTP.post('https://hooks.slack.com/...', message.to_json)

    notified = OrderNotified.build(message.stream_id)
    Sourced::Actions::AppendNext.new([notified])
  end
end
```

```ruby
def self.handle(command, history:)
  order = Order.new
  evolve(order, history)
  decide(order, command) # <= produce new events
end
```

```ruby
class Order
  include Sourced::Handler

  on StartOrder do |cmd, history:|
    # etc
    [new_message]
  end

  on AddProduct do |cmd, history:|
    # etc
    [new_message]
  end
end
```

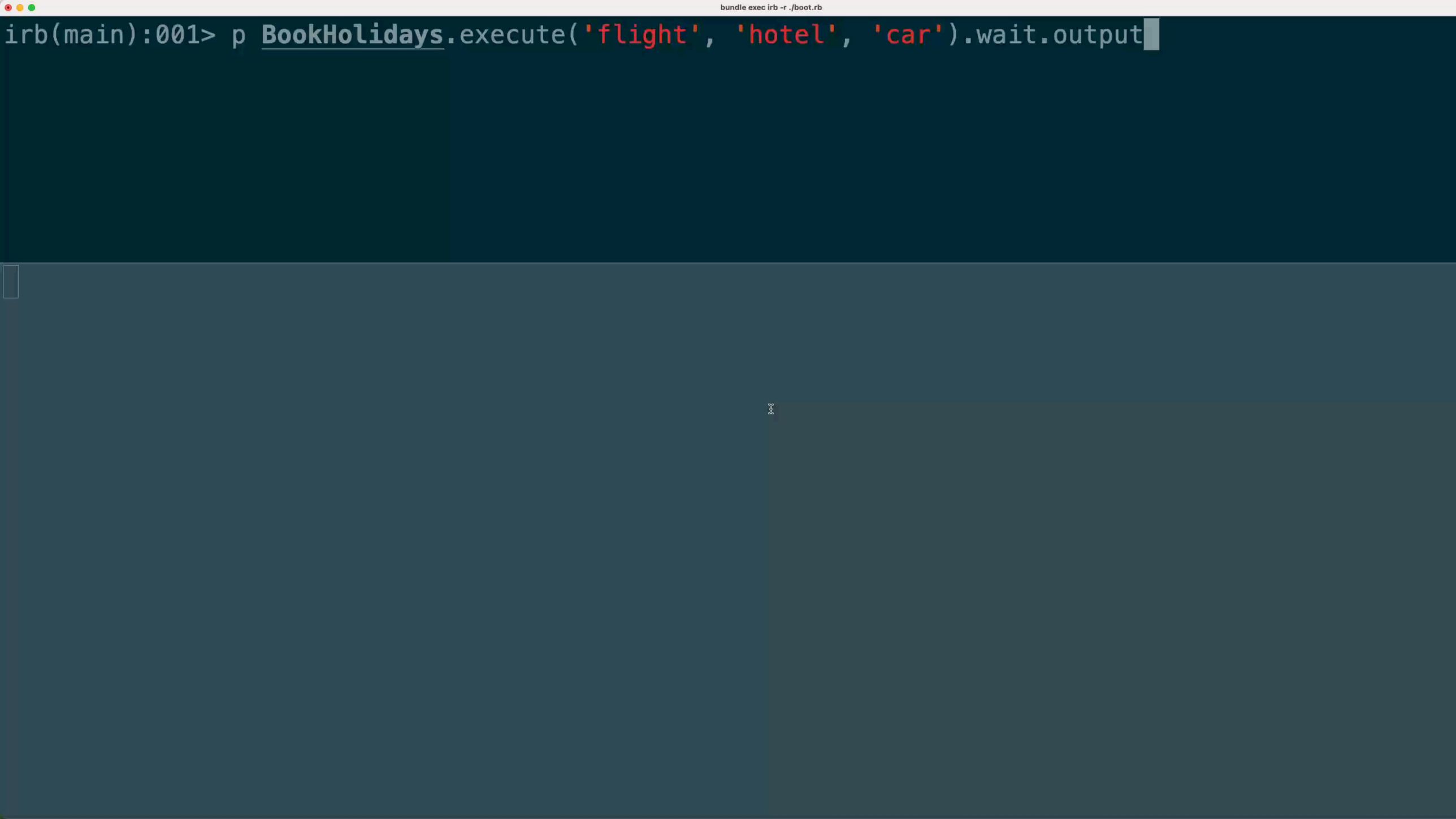
```ruby
class Order < Sourced::Actor
  state do |stream_id|
    Order.new(stream_id)
  end

  command AddProduct do |order, cmd|
    return unless order.open?

    event ProductAdded, cmd.payload
  end

  event ProductAdded do |order, event|
    order.add_product(event.payload)
  end
end
```

```ruby
class Order < Sourced::Actor
  # ...

  event PaymentStarted do |order, event|
    order.payment.status = :started
  end

  reaction PaymentStarted do |order, event|
    dispatch(Payment::StartPayment, order.payment).to("#{order.id}-payment")
  end
end
```


Time
Start Payment
"order-1"
Payment started
reaction
Confirm payment
Payment confirmed
reaction
Payments API
"order-1-payment"
Start Payment
Payment started

Durable execution

```ruby
class BookHolidays < Sourced::DurableWorkflow
  def execute(flight_info, hotel_info, car_info)
    flight_booking = book_flight(flight_info)
    hotel_booking = book_hotel(hotel_info)
    car_booking = book_car(car_info)
    confirm_booking(flight_booking, hotel_booking, car_booking)
  end

  durable def book_flight(flight_info)
    FlightsAPI.book(flight_into)
  end

  durable def book_hotel(hotel_info)
    HotelsAPI.book(hotel_info)
  end

  durable def book_car(car_info)
    CarsAPI.bool(car_info)
```


bundle exec irb -r ./boot.rb
irb(main):001> p BookHolidays.execute('flight', 'hotel', 'car').wait.output

Event correlation


```
}
}
```


8 book_holidays.step.started 2025-11-08 11:50:50 +0000


```
{
  "key": "-426133596824165456",
  "step_name": "confirm_booking",
  "args": [
    {
      "ok": true,
      "ref": "BA 143"
    },
    {
      "ok": true,
      "ref": "Radisson Riga"
    },
    {
      "ok": true,
      "ref": "Kia Sportage"
    }
  ]
}
```


9 book_holidays.step.complete 2025-11-08 11:50:52 +0000

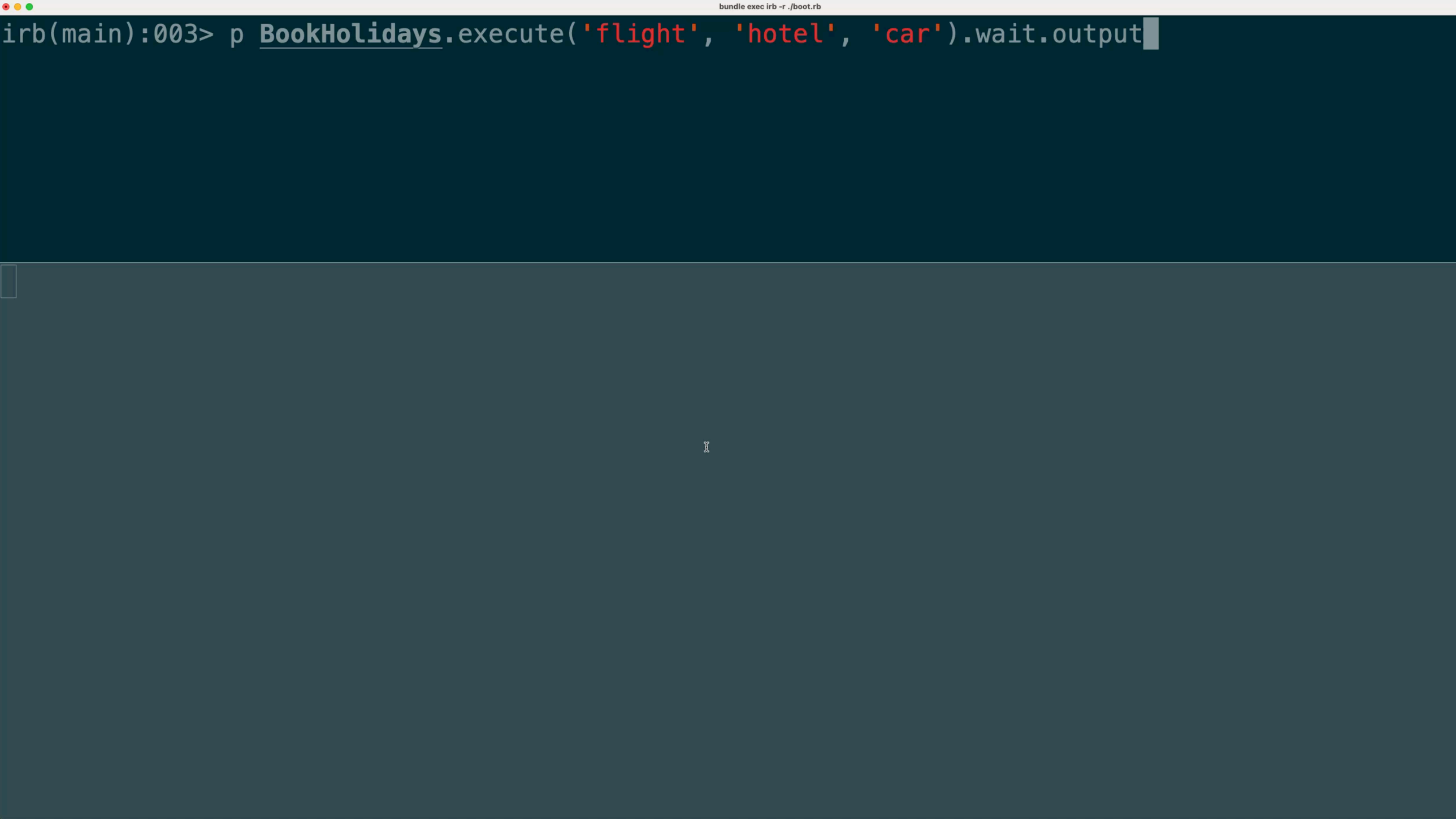

```
{
  "key": "-426133596824165456",
  "step_name": "confirm_booking",
  "output": "Booking confirmed! Flight: BA 143. Hotel: R
}
```


10 book_holidays.workflow.complete
2025-11-08 11:50:52 +0000


```
{
  "output": "Booking confirmed! Flight: BA 143. Hotel:
}
```


Close

```ruby
class HolidayLogger
  include Sourced::Handler

  on BookHolidays::StepComplete do |event|
    p [event.payload.step_name, 'complete!']
    []
  end

  on BookHolidays::WorkflowComplete do |event|
    p ['Booking complete!', event.stream_id, event.payload.output]
    []
  end
end

Sourced.register(HolidayLogger)
```


bundle exec irb -r ./boot.rb
irb(main):003> p BookHolidays.execute('flight', 'hotel', 'car').wait.output

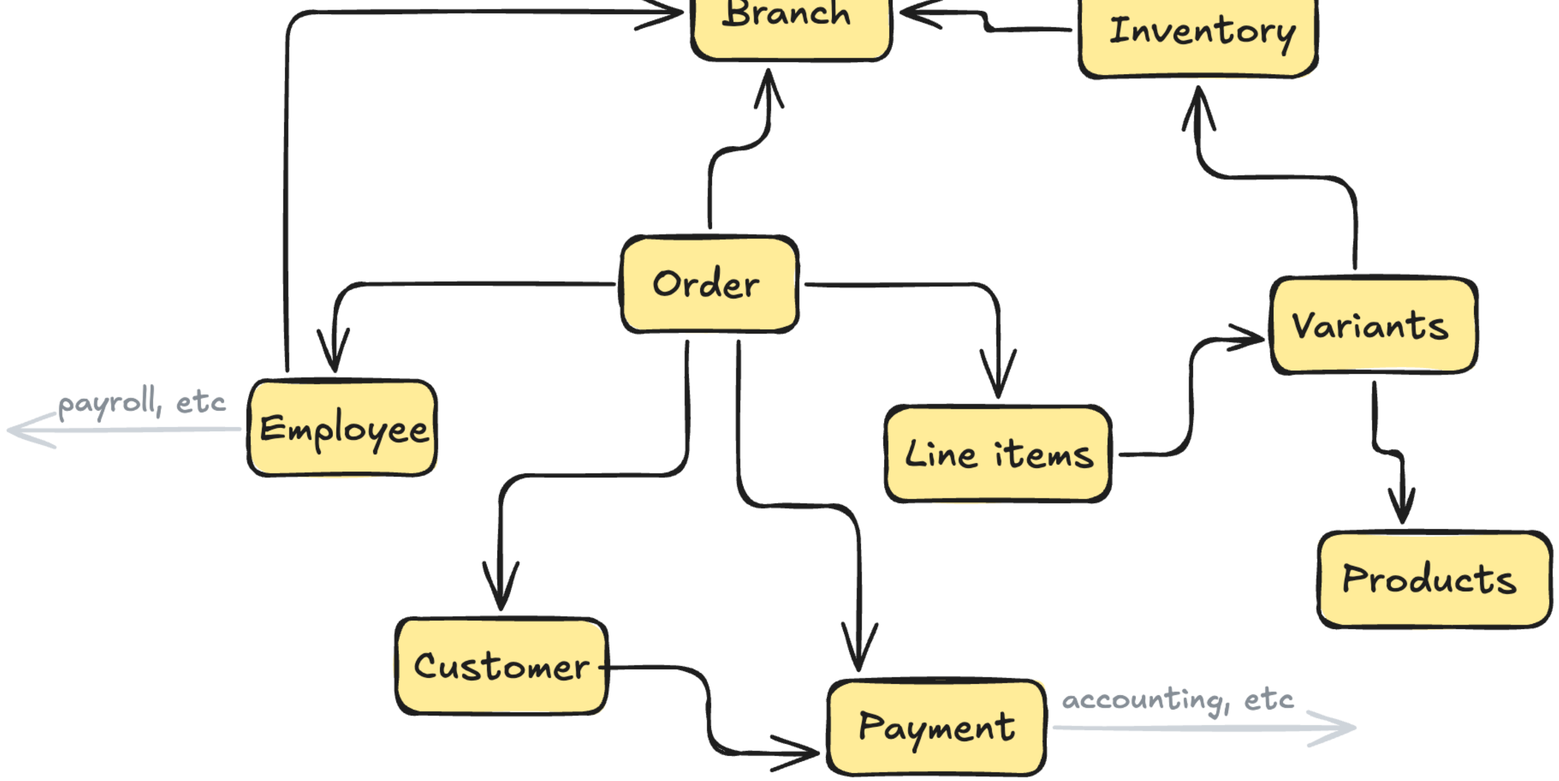
Testing

```ruby
command = Order::AddProduct.build('order-1', product_id: 'latte', …)

actions = Order.handle(command, history: [...])

expect(actions.first).to be_a(Sourced::Actions::AppendAfter)
expect(actions.first.messages).to eq([...])
```

```ruby
RSpec.describe Payment do
  subject(:payment) { Payment.new(id: 'payment-1') }

  it 'starts a payment' do
    with_reactor(payment)
      .when(Payment::Start, order_id: 'o1', amount: 1000)
      .then(Payment::Started.build(payment.id, order_id: 'o1', amount:
  end

  it 'is a no-op if payment already started' do
    with_reactor(payment)
      .given(Payment::Started, order_id: 'o1', amount: 1000)
      .when(Payment::Start, order_id: 'o1', amount: 1000)
      .then([])
  end
```

```ruby
with_reactors(Order, Payment)

  # 1. Setup current state of the world
  .given(Order::Started.build(‘order-1'))
  .given(Order::ProductAdded.build('order-1', id: 'latte', price: 100))
  .given(Order::Placed.build('order-1'))

  # 2. Dispatch new command
  .when(Order::StartPayment.build('order-1'))

  # 3. Expect new events
  .then do |_, new_messages|
    expect(new_messages).to match_sourced_messages([
      Order::StartPayment.build('order-1'),
      Order::PaymentStarted.build('order-1', amount: 100),
      Payment::Process.build('order-1-payment', amount: 100),
      Payment::Processed.build('order-1-payment', reference: 'p123')
    ])
  end
```

What this does to code

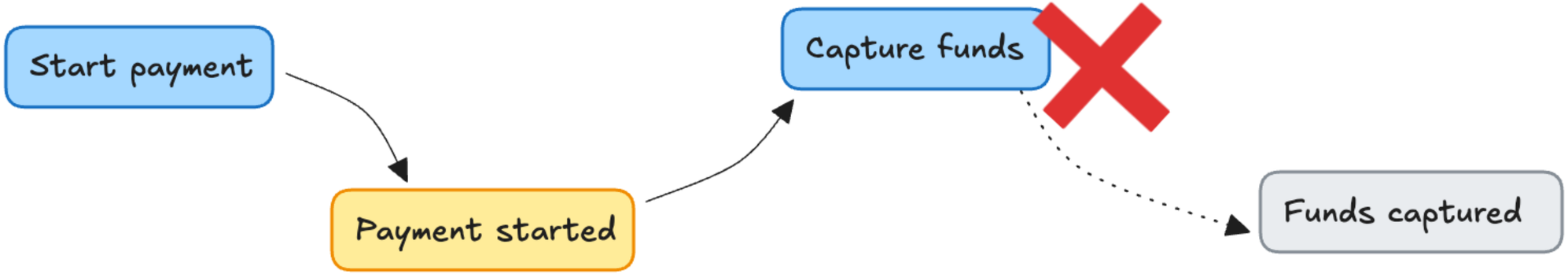

Branch
Inventory
Order
Variants
payroll, etc
Employee
Line items
Products
Customer
Payment
accounting, etc

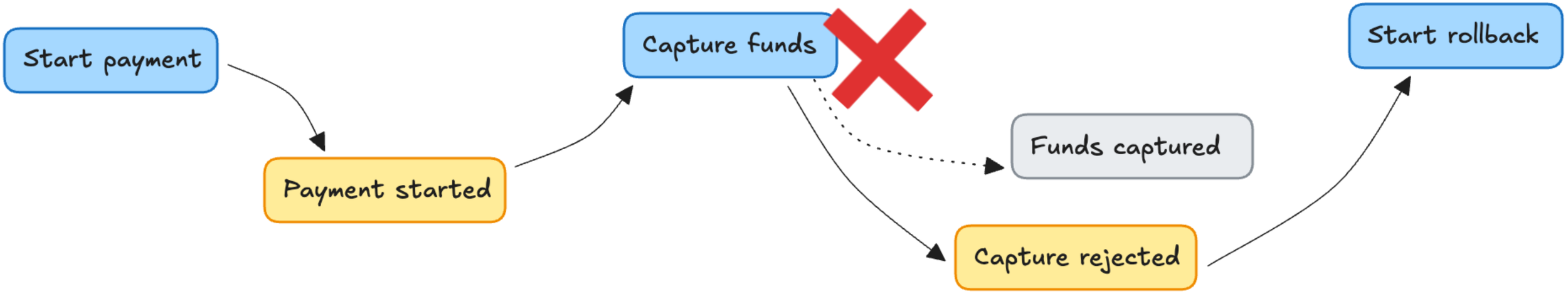

Start order
Add product
Place order

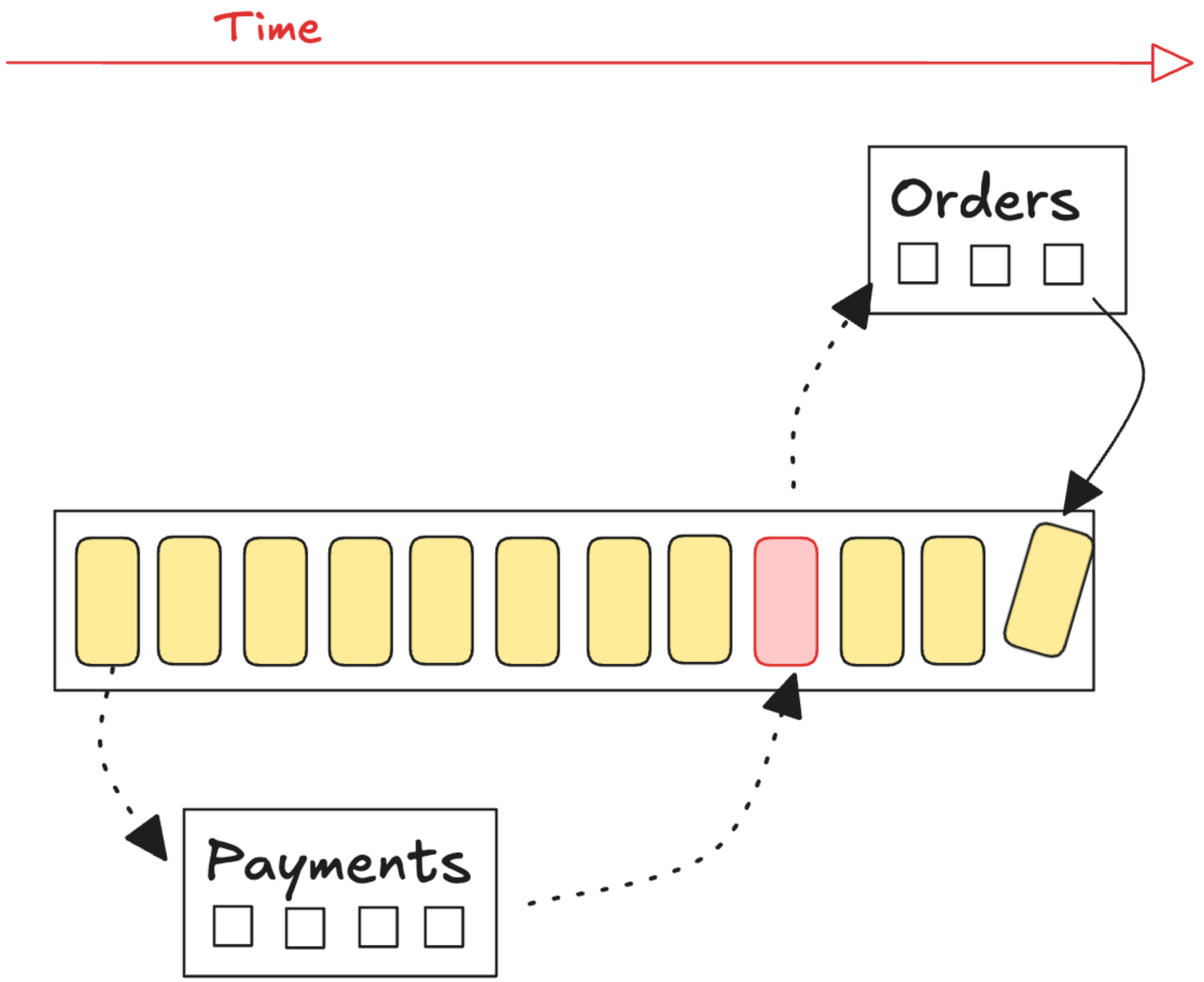
ismaelcelis.com

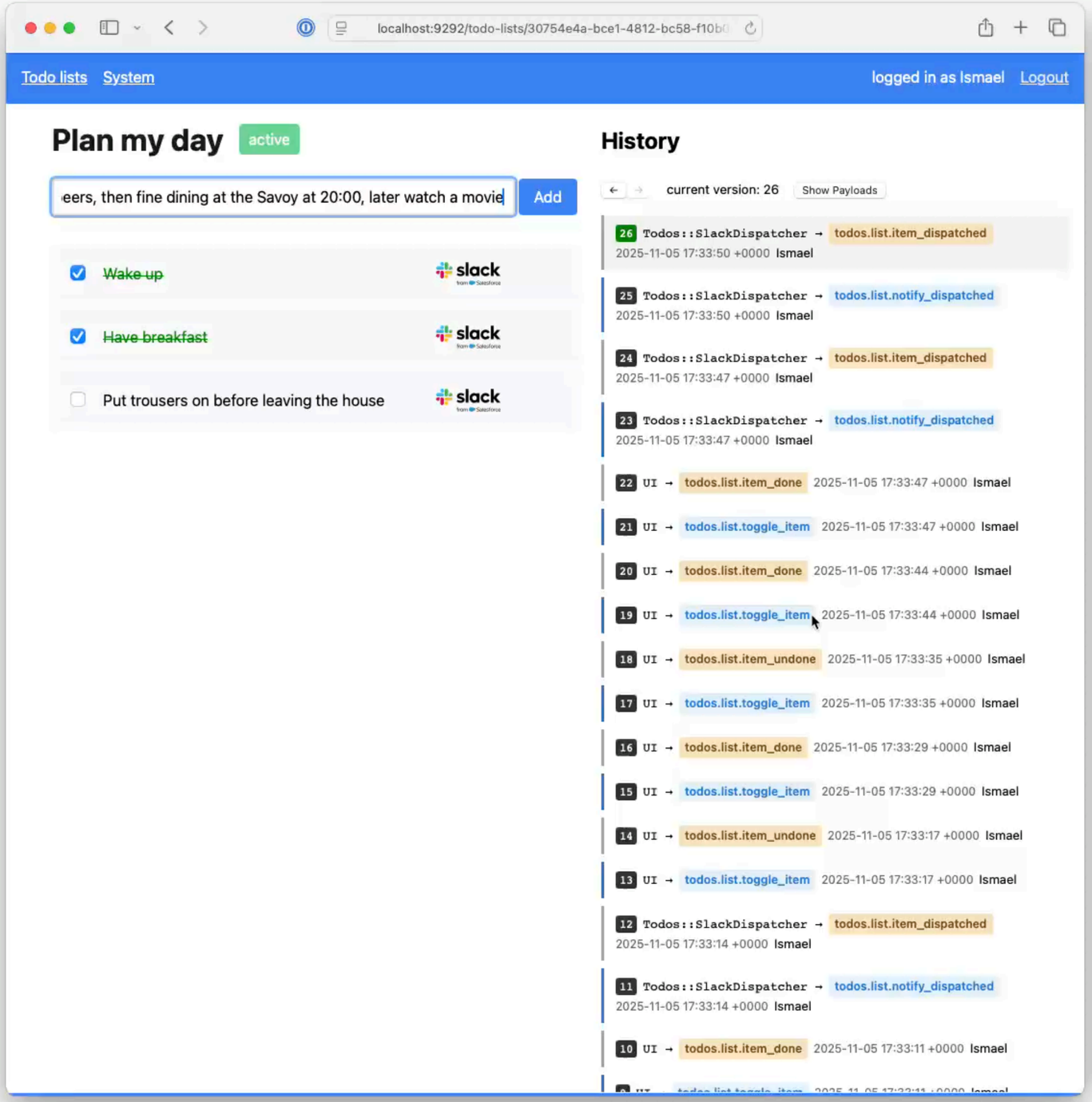
github.com/ismasan/sourced

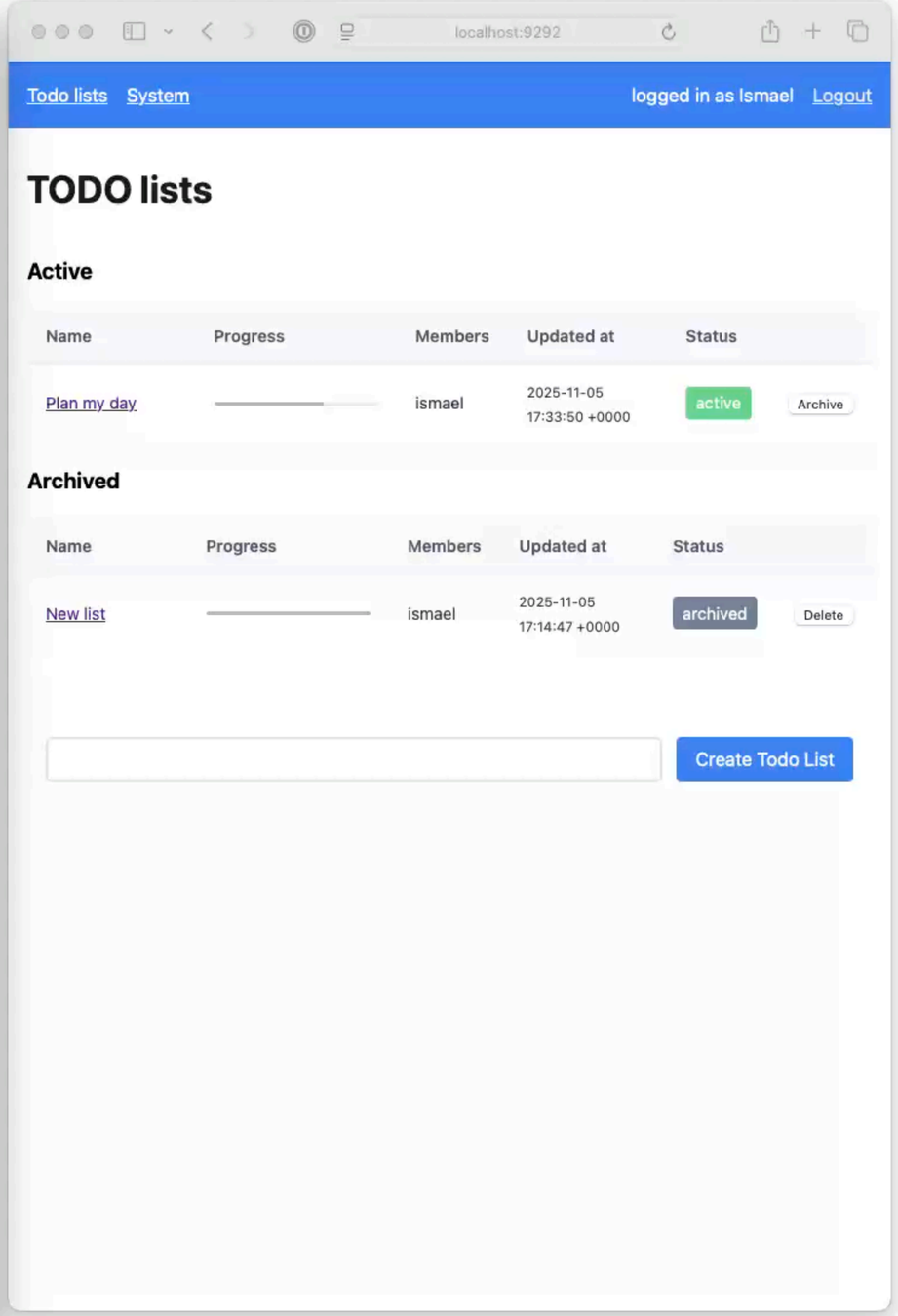
More


Start payment
Payment started
Capture funds
Funds captured

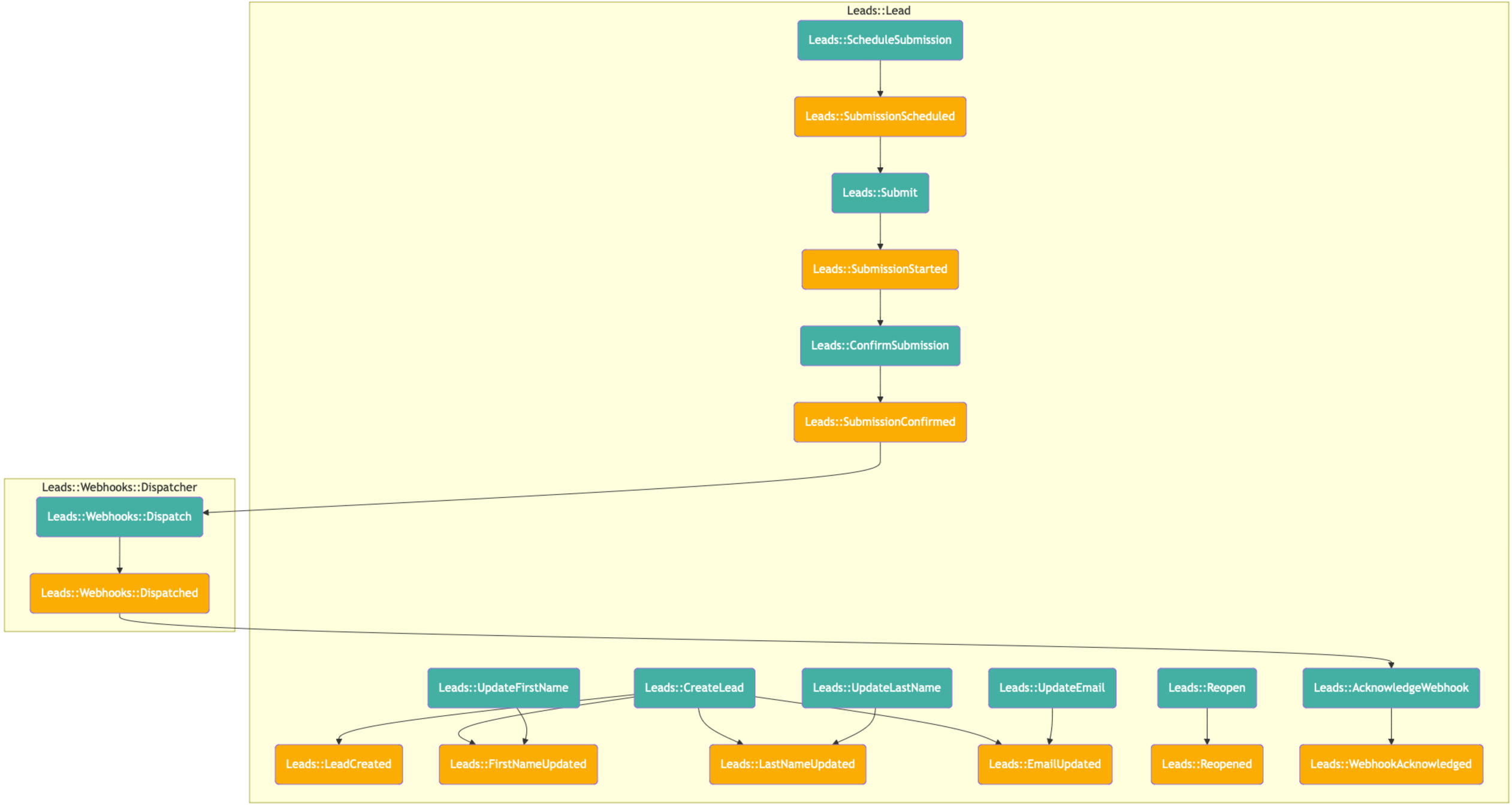

Start payment
Payment started
Capture funds
Funds captured
Capture rejected
Start rollback


Time
Orders
Payments


localhost:9292/todo-lists/30754e4a-bce1-4812-bc58-f10b0
Todo lists
System
logged in as Ismael
Logout
Plan my day
active
eers, then fine dining at the Savoy at 20:00, later watch a movie
Add
Wake up
Have breakfast
Put trousers on before leaving the house
slack
History
current version: 26
Show Payloads
26 Todos::SlackDispatcher → todos.list.item_dispatched
2025-11-05 17:33:50 +0000 Ismael
25 Todos::SlackDispatcher → todos.list.notify_dispatched
2025-11-05 17:33:50 +0000 Ismael
24 Todos::SlackDispatcher → todos.list.item_dispatched
2025-11-05 17:33:47 +0000 Ismael
23 Todos::SlackDispatcher → todos.list.notify_dispatched
2025-11-05 17:33:47 +0000 Ismael
22 UI → todos.list.item_done 2025-11-05 17:33:47 +0000 Ismael
21 UI → todos.list.toggle_item 2025-11-05 17:33:47 +0000 Ismael
20 UI → todos.list.item_done 2025-11-05 17:33:44 +0000 Ismael
19 UI → todos.list.toggle_item 2025-11-05 17:33:44 +0000 Ismael
18 UI → todos.list.item_undone 2025-11-05 17:33:35 +0000 Ismael
17 UI → todos.list.toggle_item 2025-11-05 17:33:35 +0000 Ismael
16 UI → todos.list.item_done 2025-11-05 17:33:29 +0000 Ismael
15 UI → todos.list.toggle_item 2025-11-05 17:33:29 +0000 Ismael
14 UI → todos.list.item_undone 2025-11-05 17:33:17 +0000 Ismael
13 UI → todos.list.toggle_item 2025-11-05 17:33:17 +0000 Ismael
12 Todos::SlackDispatcher → todos.list.item_dispatched
2025-11-05 17:33:14 +0000 Ismael
11 Todos::SlackDispatcher → todos.list.notify_dispatched
2025-11-05 17:33:14 +0000 Ismael
10 UI → todos.list.item_done 2025-11-05 17:33:11 +0000 Ismael


localhost:9292
Todo lists
System
logged in as Ismael
Logout
TODO lists
Active
Name
Progress
Members
Updated at
Status
Plan my day
ismael
2025-11-05 17:33:50 +0000
active
Archive
Archived
Name
Progress
Members
Updated at
Status
New list
ismael
2025-11-05 17:14:47 +0000
archived
Delete
Create Todo List

Flow


Leads::Lead
Leads::ScheduleSubmission
Leads::SubmissionScheduled
Leads::Submit
Leads::SubmissionStarted
Leads::ConfirmSubmission
Leads::SubmissionConfirmed
Leads::Webhooks::Dispatcher
Leads::Webhooks::Dispatch
Leads::Webhooks::Dispatched
Leads::UpdateFirstName
Leads::CreateLead
Leads::UpdateLastName
Leads::UpdateEmail
Leads::Reopen
Leads::AcknowledgeWebhook
Leads::LeadCreated
Leads::FirstNameUpdated
Leads::LastNameUpdated
Leads::EmailUpdated
Leads::Reopened
Leads::WebhookAcknowledged

```ruby
class BookHolidays < Sourced::DurableWorkflow
  def execute(flight_info, hotel_info, car_info)
    flight_booking = book_flight(flight_info)
    hotel_booking = book_hotel(hotel_info)

    wait 10 # <==

    car_booking = book_car(car_info)
    confirm_booking(flight_booking, …)
  end

end
```