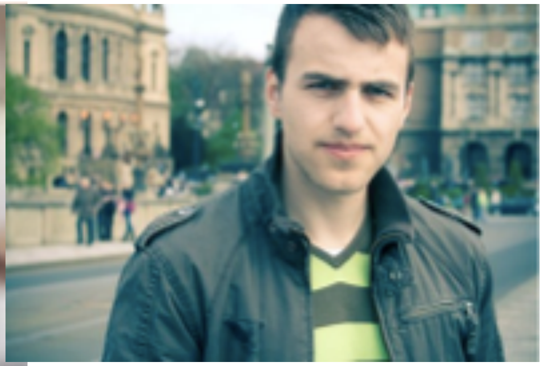
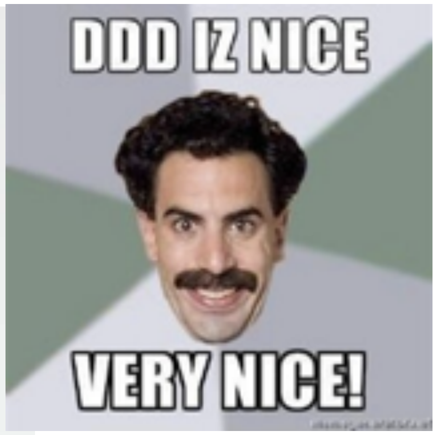
Event-Driven Design

Rinat Abdullin | abdullin.com | @abdullin

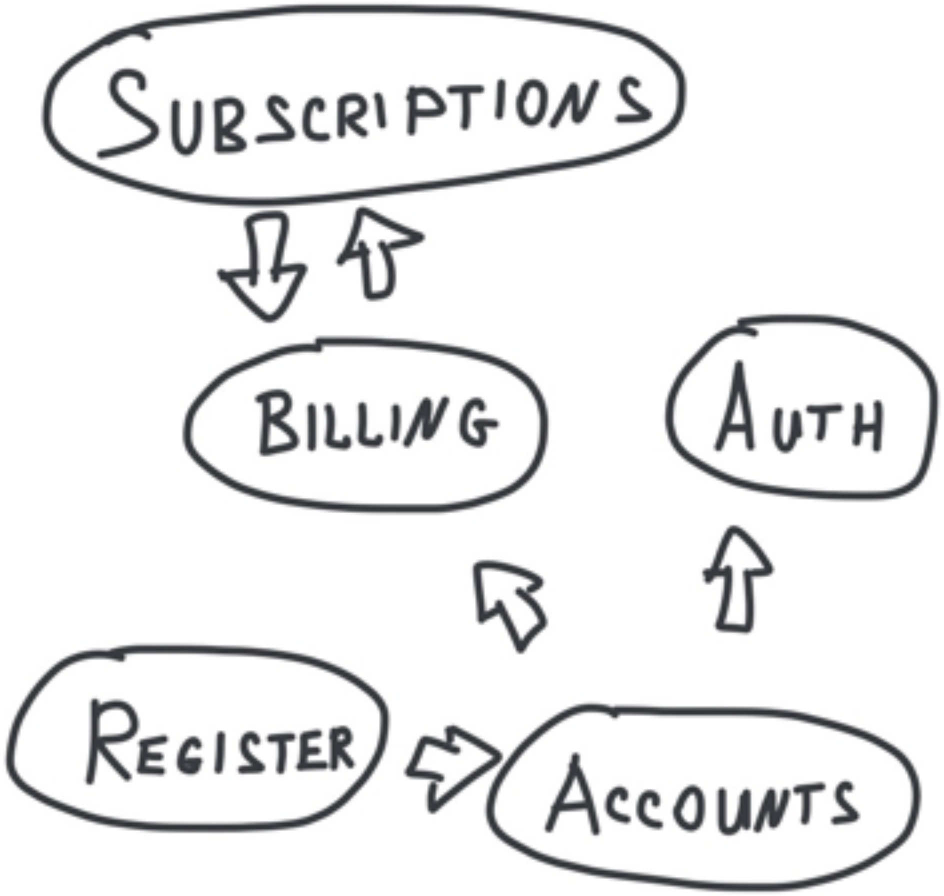
Plan

- Software Design
- Event Storming
- Practical Event-Driven Design
- CQRS Beers

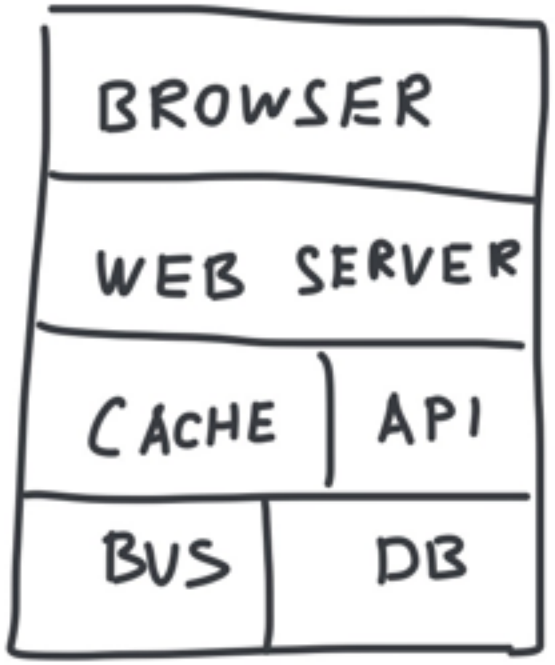



DDD IZ NICE
VERY NICE!

Software Design Process



VS.



Software Design

Divide and Conquer

Context Map

Map is not the territory

“First make it possible. Then make it beautiful. Then make it fast.”

–Nathan Marz, Suffering-oriented programming

Feedback Loops

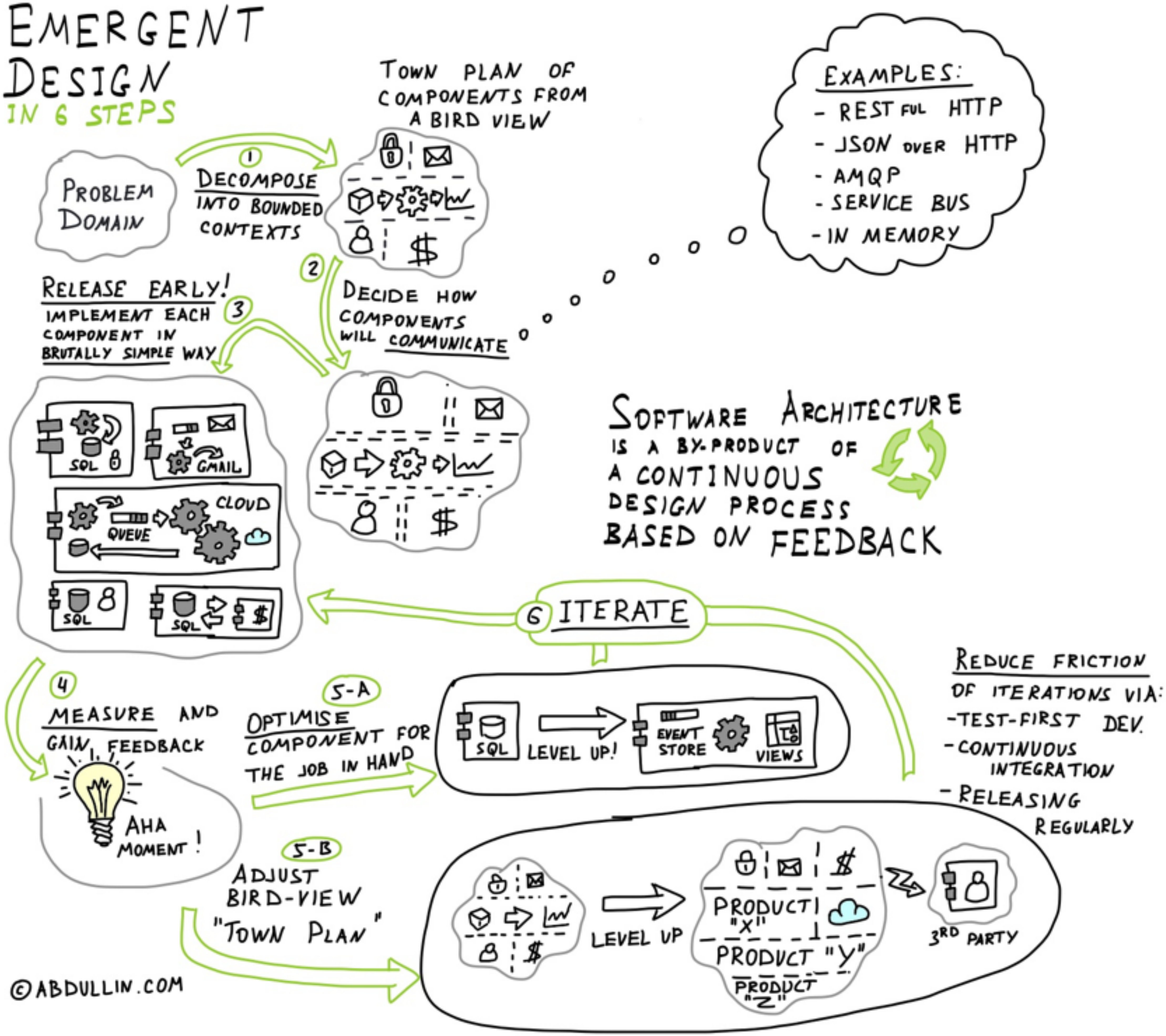
“Software development is a **learning process**.

Working code is a side-effect.”

–EU DDD Community

Iterate & Reduce Friction

EMERGENT DESIGN IN 6 STEPS

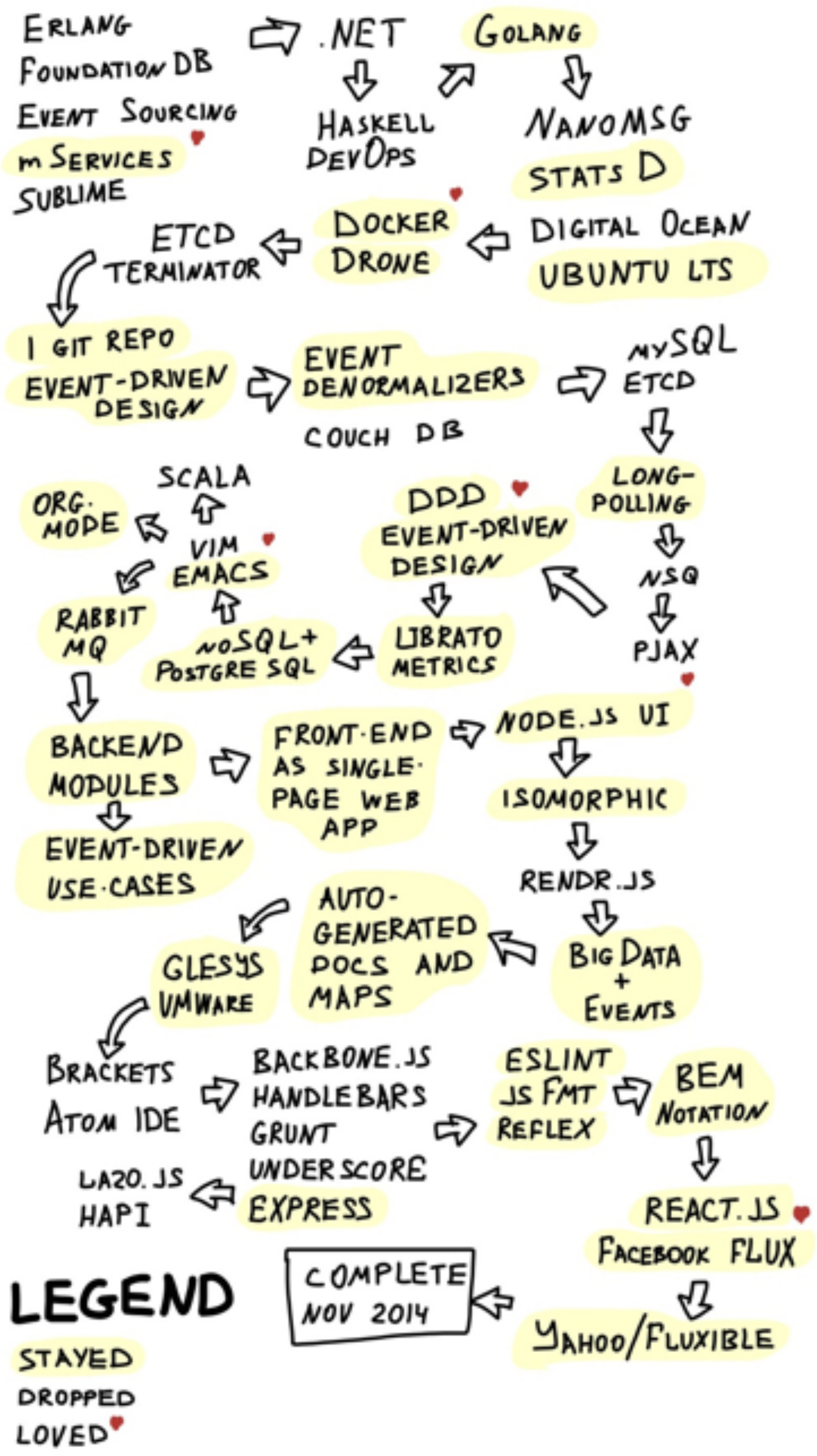

PROBLEM DOMAIN
1 DECOMPOSE INTO BOUNDED CONTEXTS
TOWN PLAN OF COMPONENTS FROM A BIRD VIEW
2 DECIDE HOW COMPONENTS WILL COMMUNICATE
EXAMPLES:
- RESTFUL HTTP
- JSON OVER HTTP
- AMQP
- SERVICE BUS
- IN MEMORY
RELEASE EARLY! 3 IMPLEMENT EACH COMPONENT IN BRUTALLY SIMPLE WAY
SQL
GMAIL
CLOUD
QUEUE
SQL
SQL
6 ITERATE
4 MEASURE AND GAIN FEEDBACK
AHA MOMENT!
5-A OPTIMISE COMPONENT FOR THE JOB IN HAND
SQL
LEVEL UP!
EVENT STORE
VIEWS
REDUCE FRICTION OF ITERATIONS VIA:
-TEST-FIRST DEV.
-CONTINUOUS INTEGRATION
-RELEASING REGULARLY
5-B ADJUST BIRD-VIEW "TOWN PLAN"
LEVEL UP
PRODUCT "X"
PRODUCT "Y"
PRODUCT "Z"
3RD PARTY

SOFTWARE ARCHITECTURE IS A BY-PRODUCT OF A CONTINUOUS DESIGN PROCESS BASED ON FEEDBACK

©ABDULLIN.COM

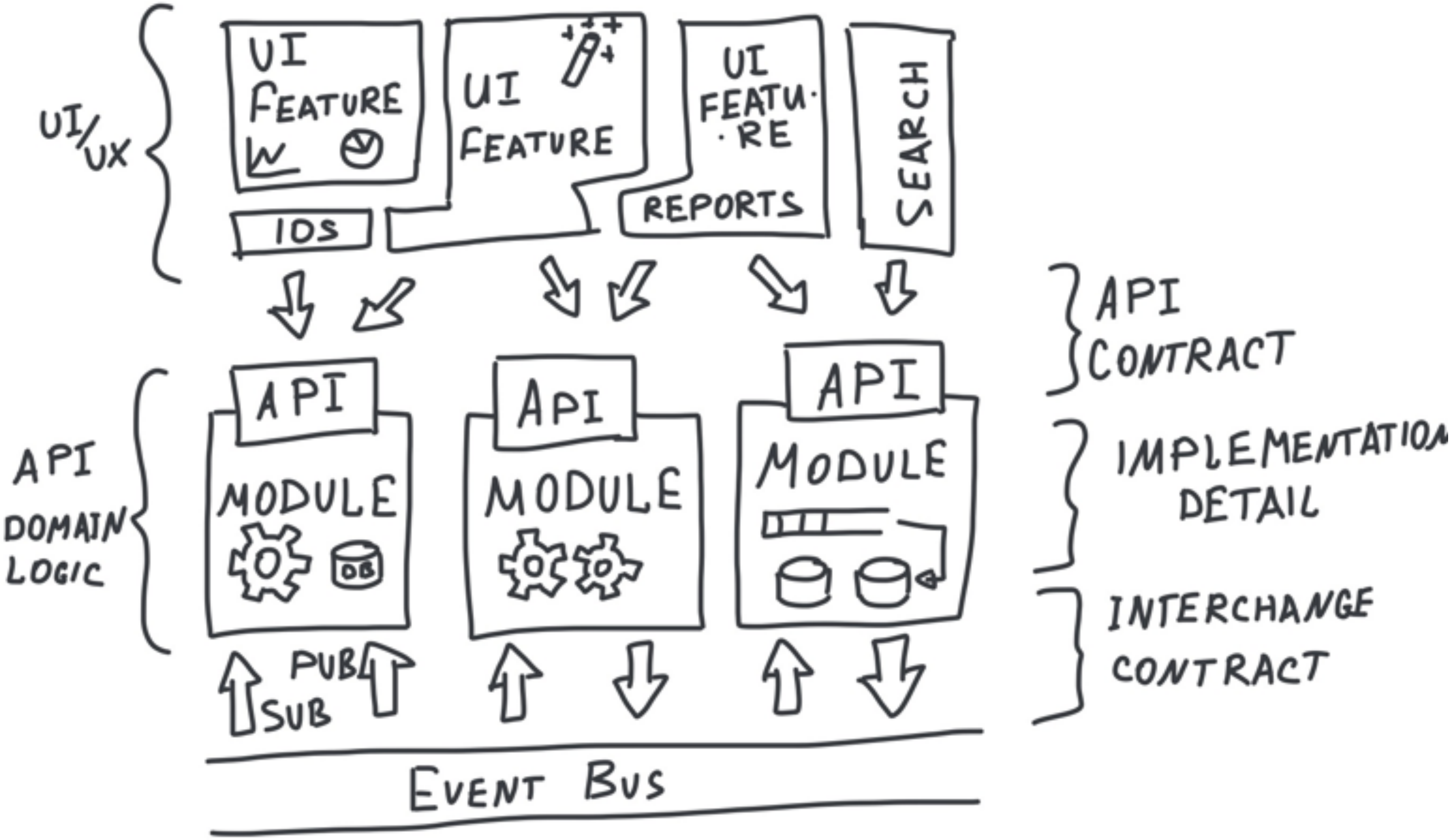
One Example

Rapid Iterations FTW



Events and API

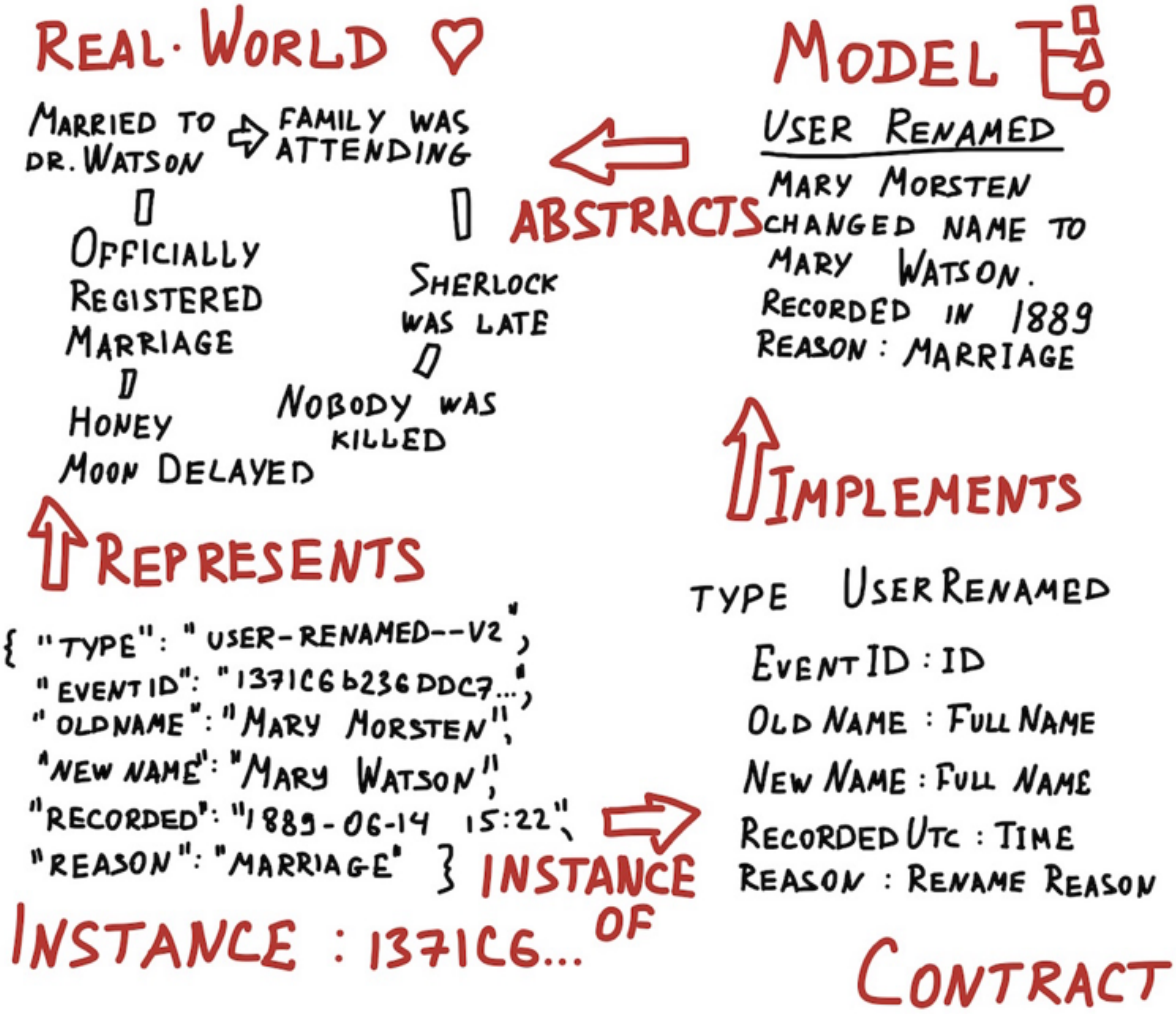
Interchange Context + ACL



Contracts at Boundary

Events, API, ACL

MANY FACES OF DOMAIN EVENT



MARRIED TO DR. WATSON ⇒ FAMILY WAS ATTENDING

⇓

OFFICIALLY REGISTERED MARRIAGE

⇓

HONEY MOON DELAYED

⇓

SHERLOCK WAS LATE

⇓

NOBODY WAS KILLED

USER RENAMED

MARY MORSTEN CHANGED NAME TO MARY WATSON.
RECORDED IN 1889
REASON : MARRIAGE

```
{ "TYPE": "USER-RENAMED--V2",
  "EVENTID": "1371C6b236DDC7...",
  "OLDNAME": "MARY MORSTEN",
  "NEW NAME": "MARY WATSON",
  "RECORDED": "1889-06-14 15:22",
  "REASON": "MARRIAGE" }
```

```
TYPE USERRENAMED
  EVENTID : ID
  OLD NAME : FULL NAME
  NEW NAME : FULL NAME
  RECORDEDUTC : TIME
  REASON : RENAME REASON
```

Questions?

Event-Storming

Let’s explore our boundaries

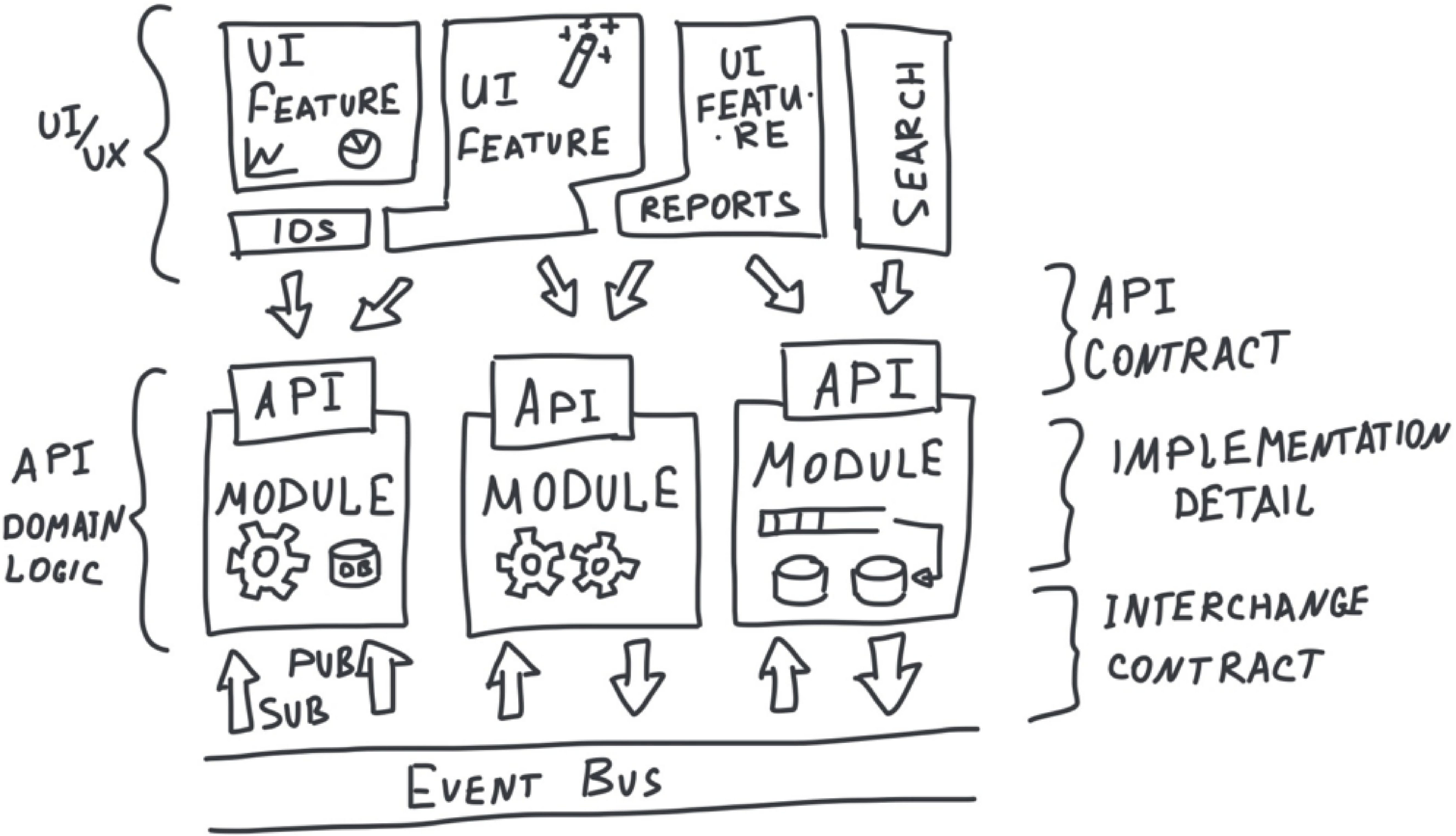
Requirements

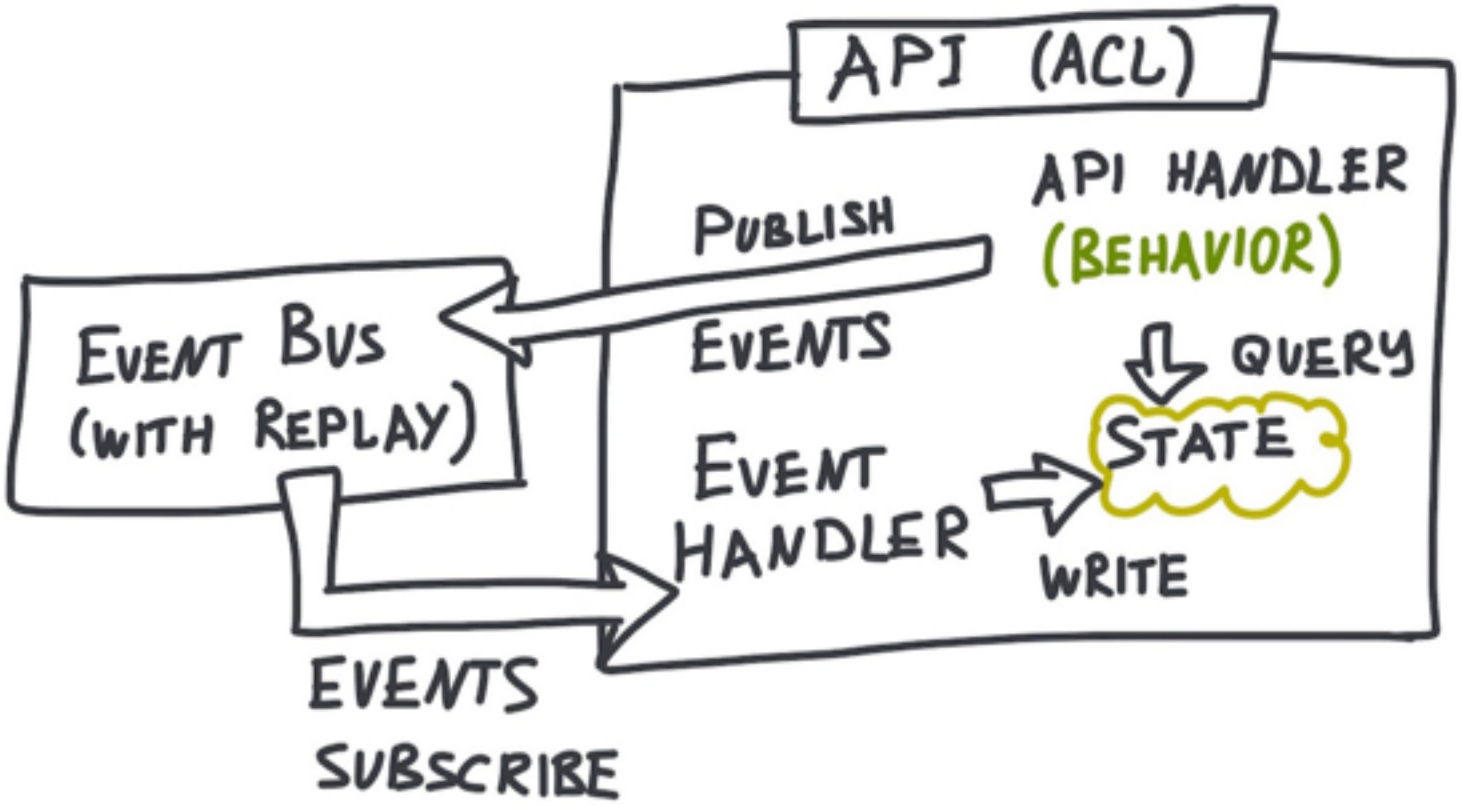
- Invite domain experts and developers
- No chairs
- Lots of writing space
- Post-it notes and markers
- Just map the commands/events
- Everybody participates
- Have fun

"IT'S LIKE T

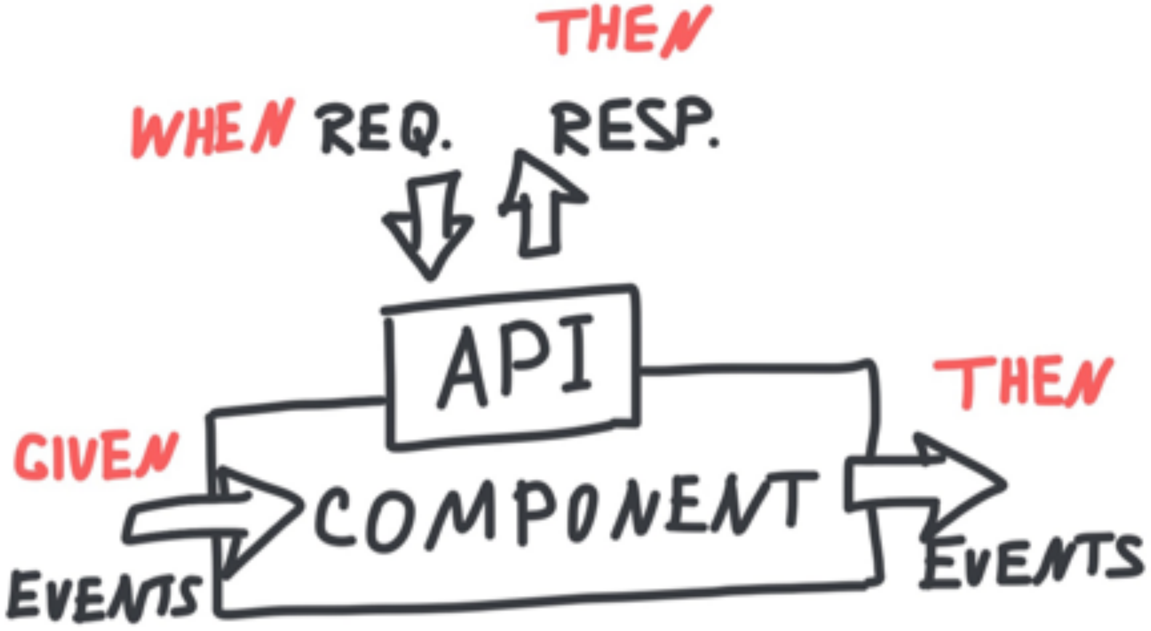
Practical Applications

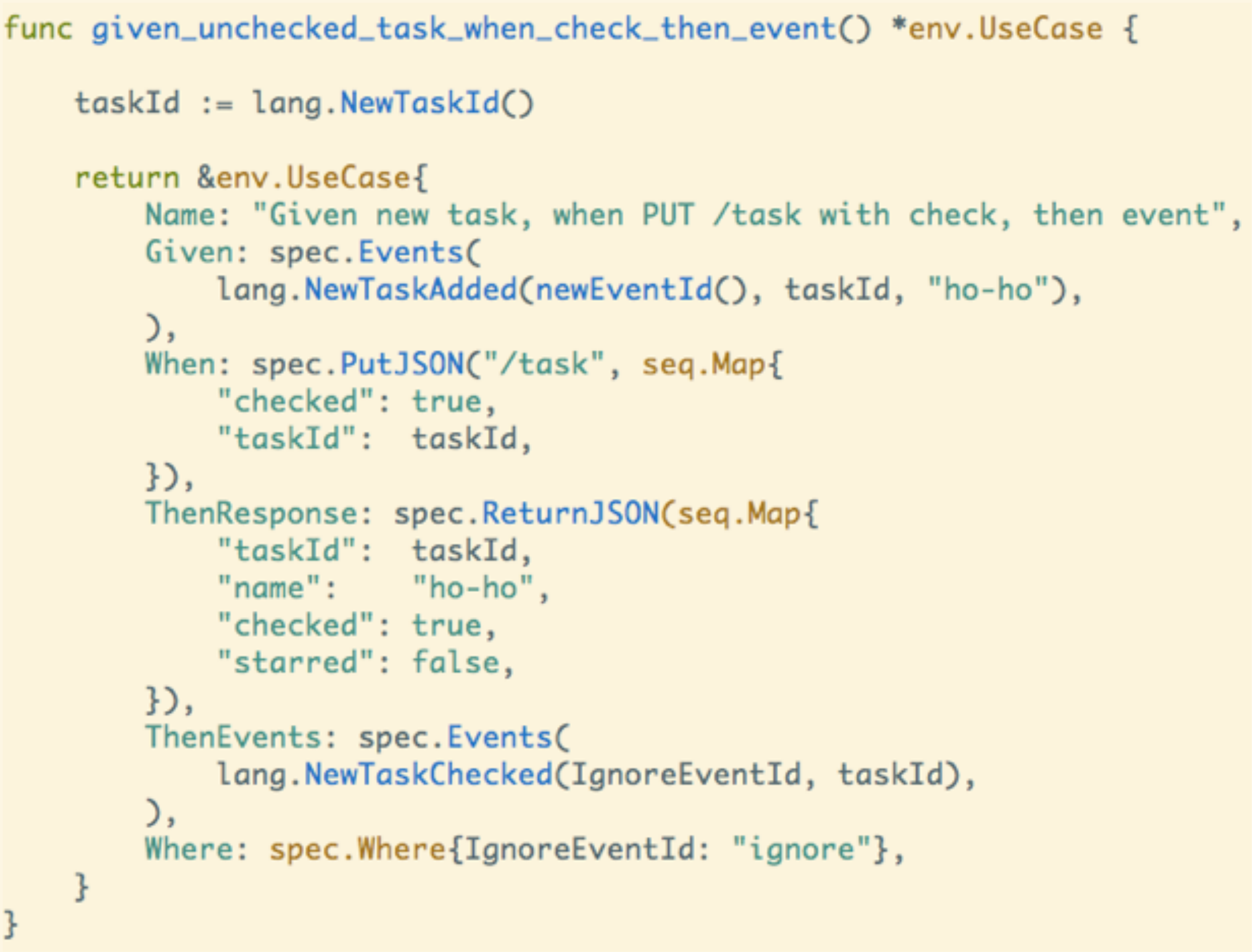
Back to the code


UI/UX
UI FEATURE
IOS
UI FEATURE
UI FEATU·RE
REPORTS
SEARCH
API CONTRACT
API
API
API
MODULE
MODULE
MODULE
DB
API DOMAIN LOGIC
IMPLEMENTATION DETAIL
INTERCHANGE CONTRACT
PUB
SUB
EVENT BUS



Capture Behaviors





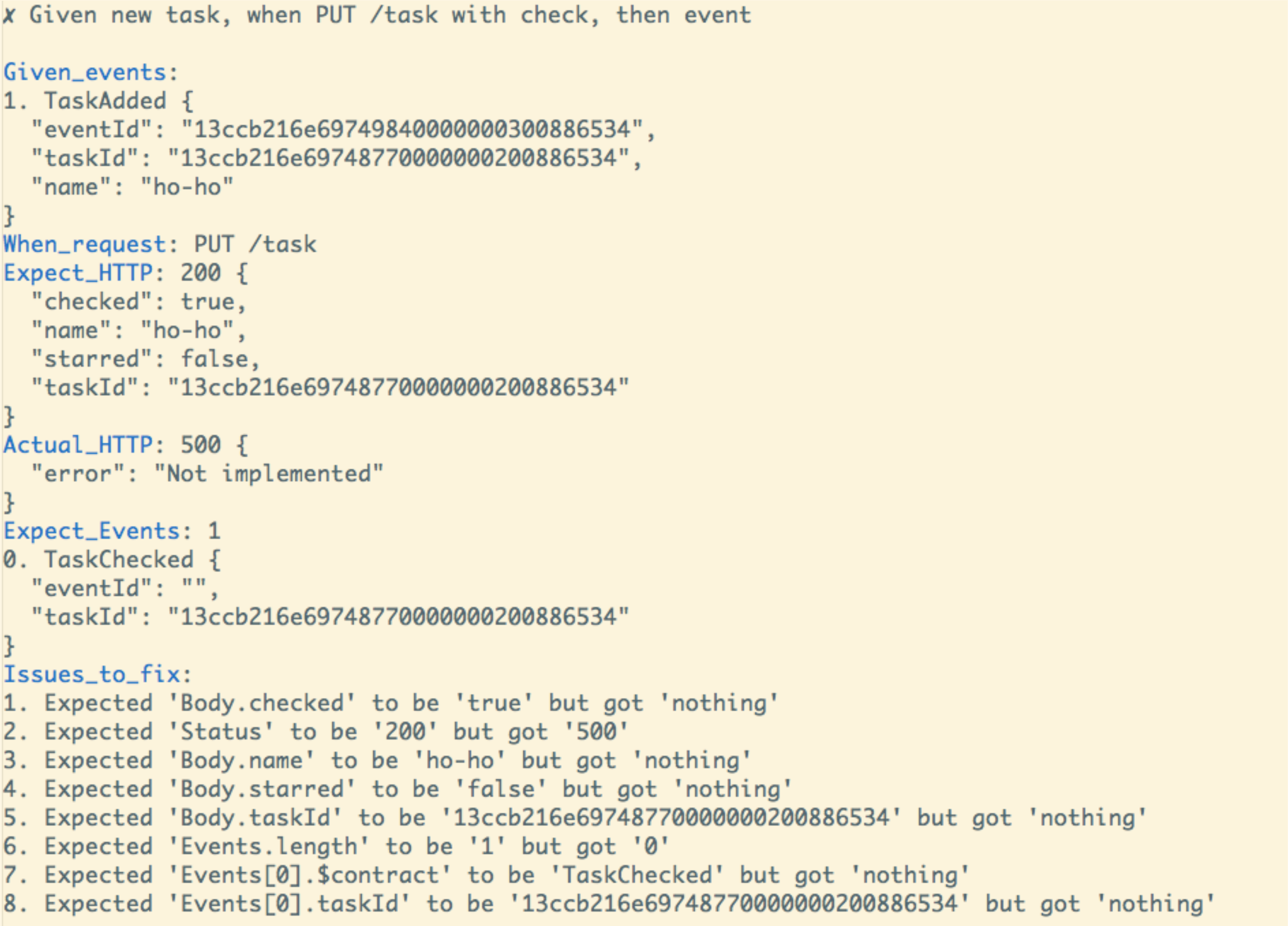
```
func given_unchecked_task_when_check_then_event() *env.UseCase {

    taskId := lang.NewTaskId()

    return &env.UseCase{
        Name: "Given new task, when PUT /task with check, then event",
        Given: spec.Events(
            lang.NewTaskAdded(newEventId(), taskId, "ho-ho"),
        ),
        When: spec.PutJSON("/task", seq.Map{
            "checked": true,
            "taskId":  taskId,
        }),
        ThenResponse: spec.ReturnJSON(seq.Map{
            "taskId":  taskId,
            "name":    "ho-ho",
            "checked": true,
            "starred": false,
        }),
        ThenEvents: spec.Events(
            lang.NewTaskChecked(IgnoreEventId, taskId),
        ),
        Where: spec.Where{IgnoreEventId: "ignore"},
    }
}
```

API Use Case

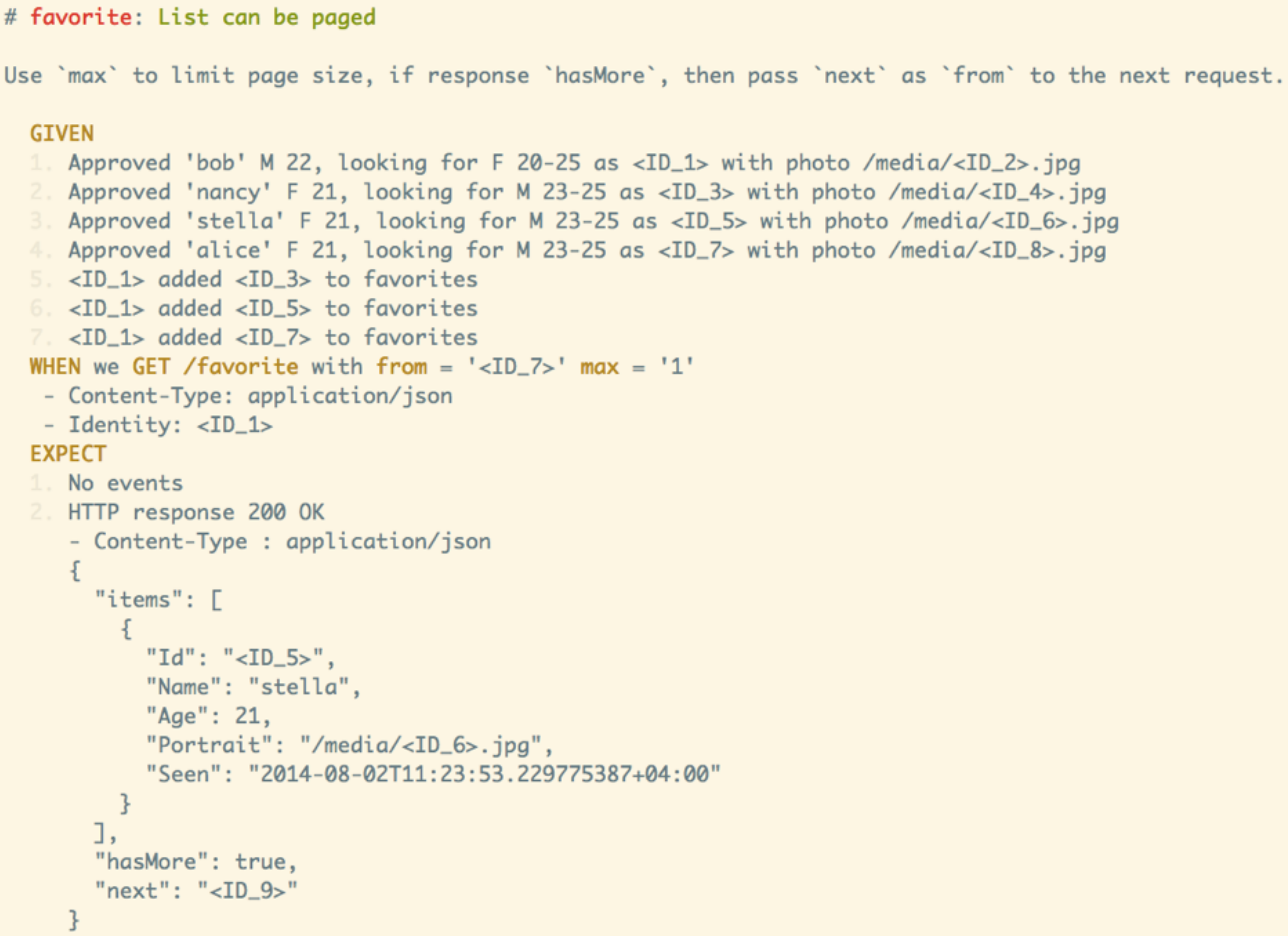
http://github.com/abdullin/omni/

Use cases

- Verify API correctness
- Allow iterative TDD/BDD
- Automatic stress-testing
- Human-readable documentation
- API documentation with samples
- Dependency diagrams
- Module sanity checks



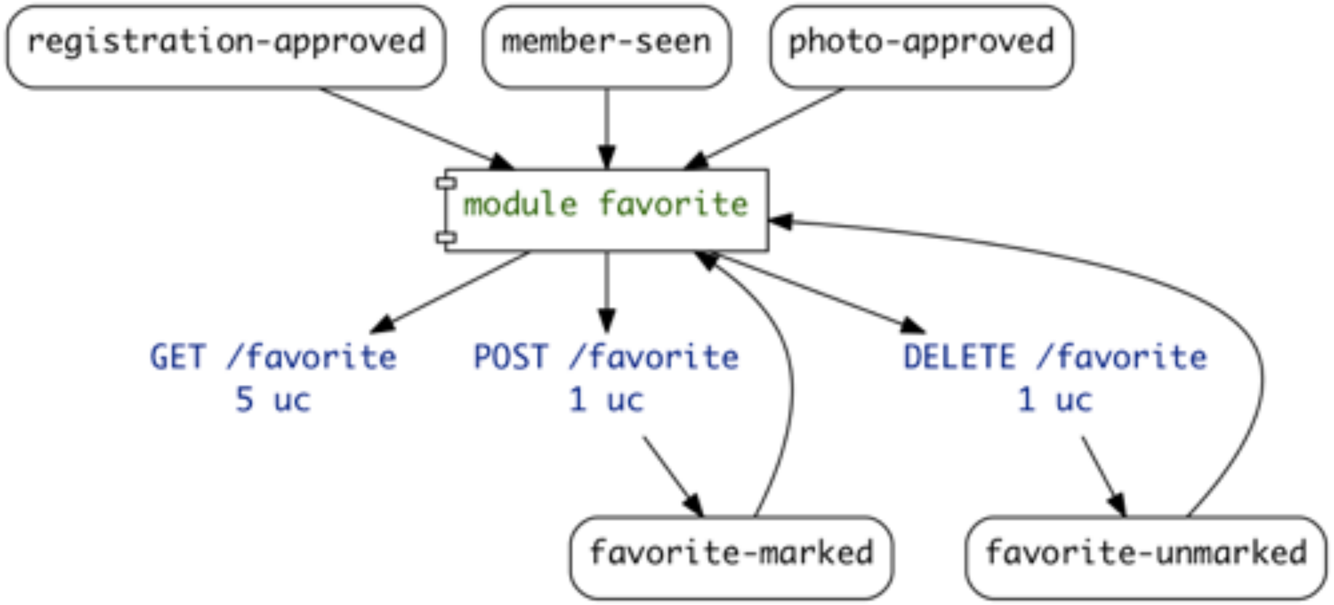
```
✗ Given new task, when PUT /task with check, then event

Given_events:
1. TaskAdded {
  "eventId": "13ccb216e69749840000000300886534",
  "taskId": "13ccb216e69748770000000200886534",
  "name": "ho-ho"
}
When_request: PUT /task
Expect_HTTP: 200 {
  "checked": true,
  "name": "ho-ho",
  "starred": false,
  "taskId": "13ccb216e69748770000000200886534"
}
Actual_HTTP: 500 {
  "error": "Not implemented"
}
Expect_Events: 1
0. TaskChecked {
  "eventId": "",
  "taskId": "13ccb216e69748770000000200886534"
}
Issues_to_fix:
1. Expected 'Body.checked' to be 'true' but got 'nothing'
2. Expected 'Status' to be '200' but got '500'
3. Expected 'Body.name' to be 'ho-ho' but got 'nothing'
4. Expected 'Body.starred' to be 'false' but got 'nothing'
5. Expected 'Body.taskId' to be '13ccb216e69748770000000200886534' but got 'nothing'
6. Expected 'Events.length' to be '1' but got '0'
7. Expected 'Events[0].$contract' to be 'TaskChecked' but got 'nothing'
8. Expected 'Events[0].taskId' to be '13ccb216e69748770000000200886534' but got 'nothing'
```

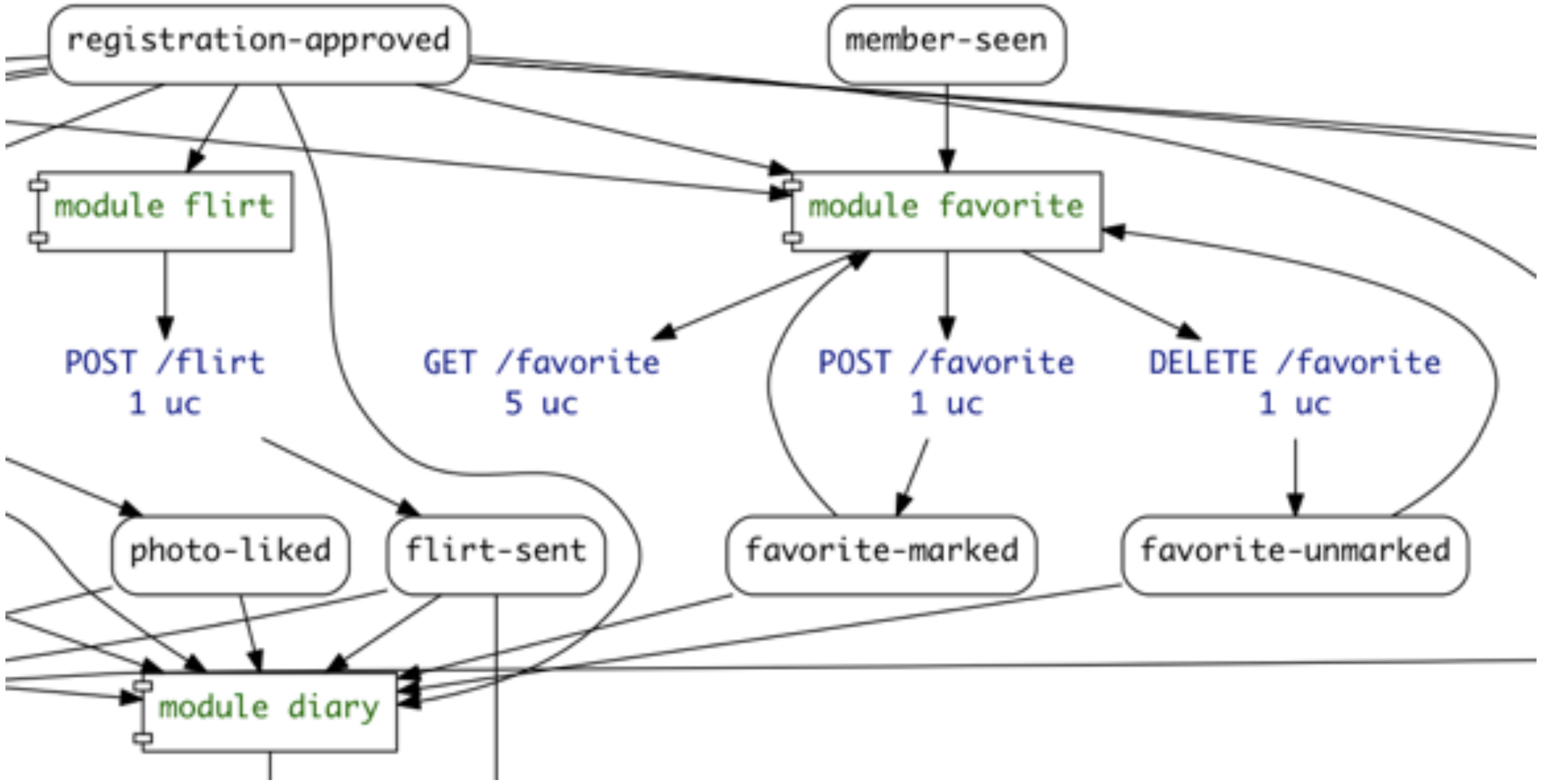


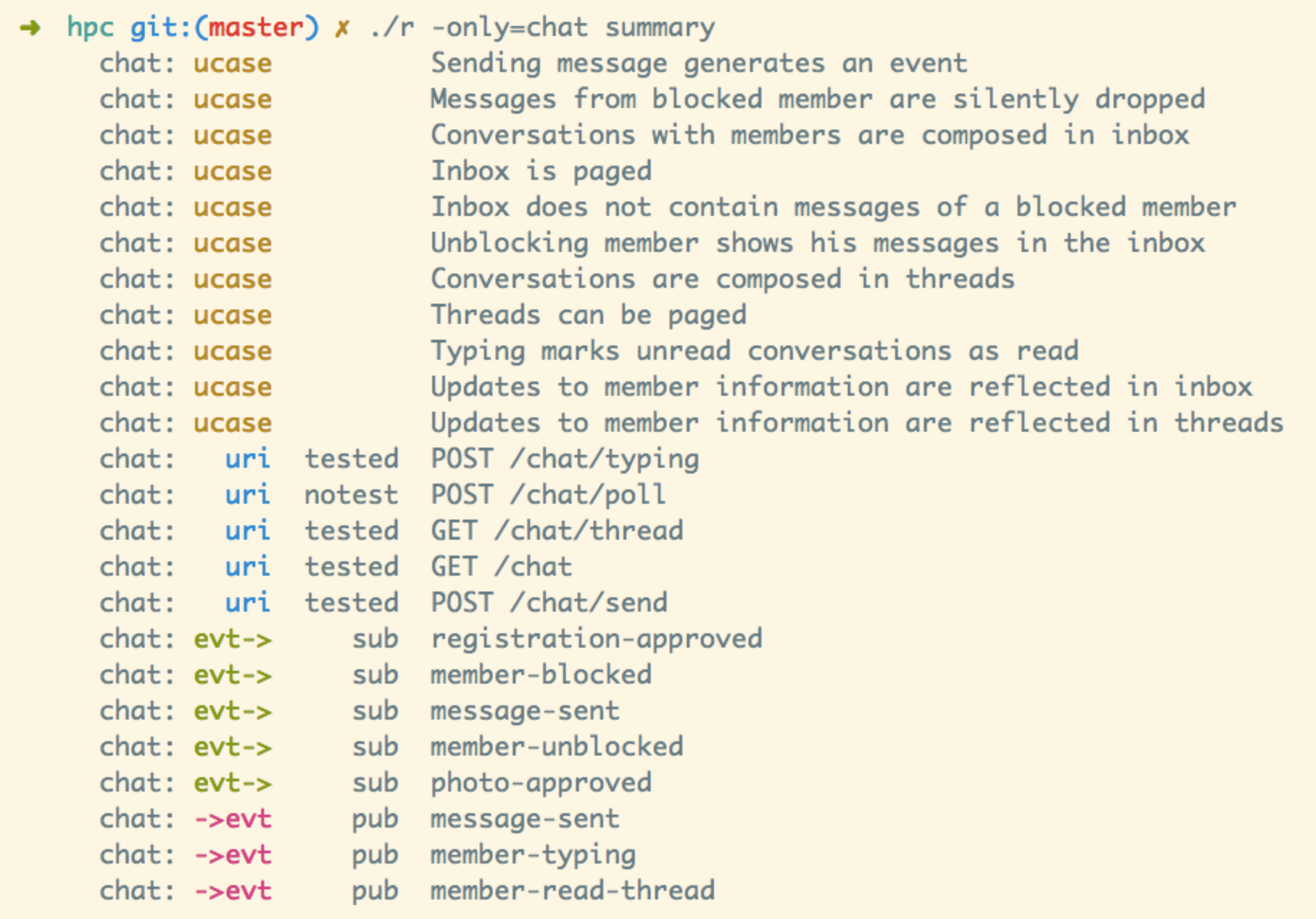
favorite: List can be paged

Use `max` to limit page size, if response `hasMore`, then pass `next` as `from` to the next request.

```
  GIVEN
  1. Approved 'bob' M 22, looking for F 20-25 as <ID_1> with photo /media/<ID_2>.jpg
  2. Approved 'nancy' F 21, looking for M 23-25 as <ID_3> with photo /media/<ID_4>.jpg
  3. Approved 'stella' F 21, looking for M 23-25 as <ID_5> with photo /media/<ID_6>.jpg
  4. Approved 'alice' F 21, looking for M 23-25 as <ID_7> with photo /media/<ID_8>.jpg
  5. <ID_1> added <ID_3> to favorites
  6. <ID_1> added <ID_5> to favorites
  7. <ID_1> added <ID_7> to favorites
  WHEN we GET /favorite with from = '<ID_7>' max = '1'
   - Content-Type: application/json
   - Identity: <ID_1>
  EXPECT
  1. No events
  2. HTTP response 200 OK
     - Content-Type : application/json
     {
       "items": [
         {
           "Id": "<ID_5>",
           "Name": "stella",
           "Age": 21,
           "Portrait": "/media/<ID_6>.jpg",
           "Seen": "2014-08-02T11:23:53.229775387+04:00"
         }
       ],
       "hasMore": true,
       "next": "<ID_9>"
     }
```


registration-approved
member-seen
photo-approved
module favorite
GET /favorite
5 uc
POST /favorite
1 uc
DELETE /favorite
1 uc
favorite-marked
favorite-unmarked


registration-approved
member-seen
module flirt
module favorite
POST /flirt
1 uc
GET /favorite
5 uc
POST /favorite
1 uc
DELETE /favorite
1 uc
photo-liked
flirt-sent
favorite-marked
favorite-unmarked
module diary

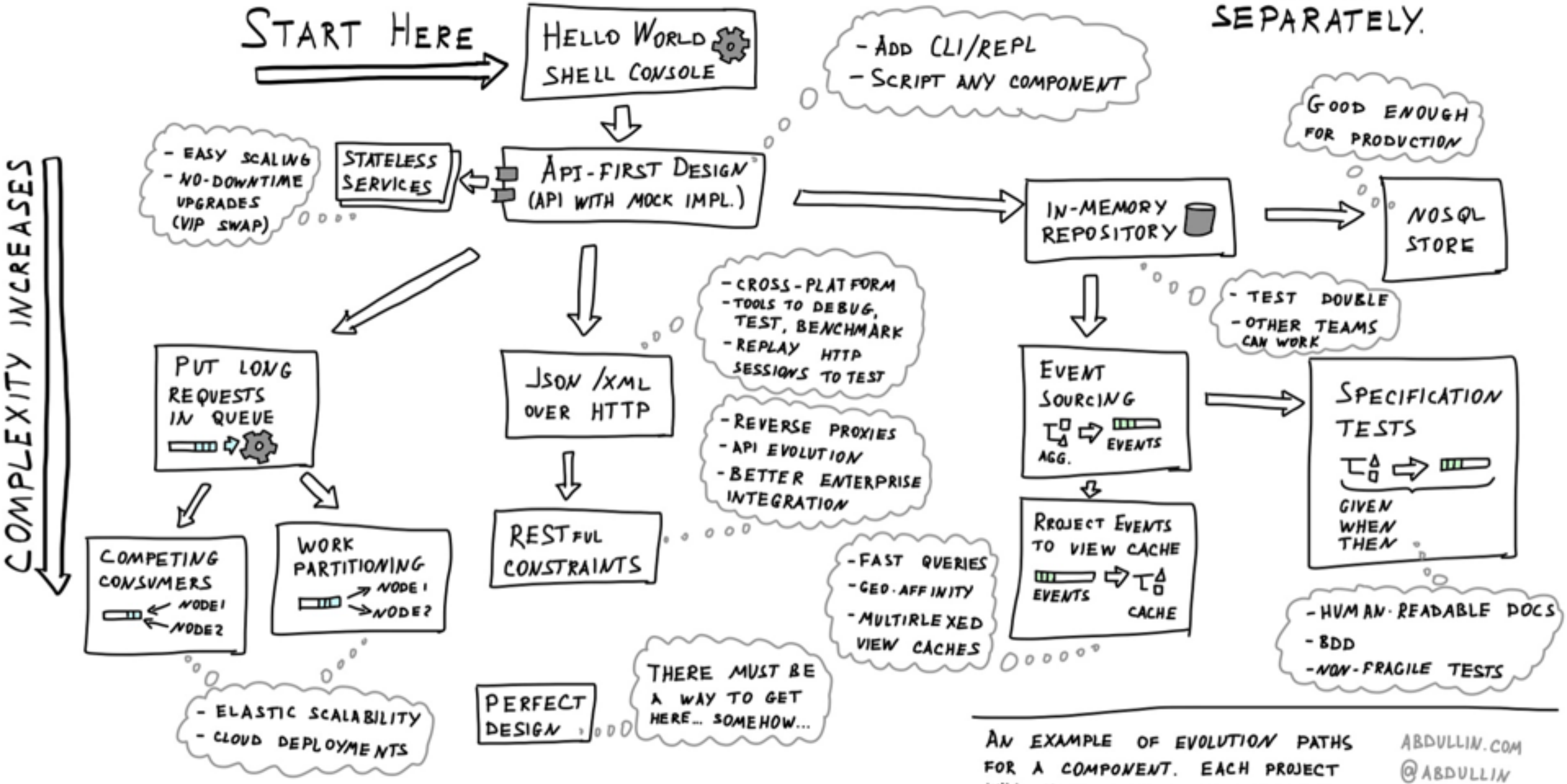


```
➜  hpc git:(master) ✗ ./r -only=chat summary
     chat: ucase          Sending message generates an event
     chat: ucase          Messages from blocked member are silently dropped
     chat: ucase          Conversations with members are composed in inbox
     chat: ucase          Inbox is paged
     chat: ucase          Inbox does not contain messages of a blocked member
     chat: ucase          Unblocking member shows his messages in the inbox
     chat: ucase          Conversations are composed in threads
     chat: ucase          Threads can be paged
     chat: ucase          Typing marks unread conversations as read
     chat: ucase          Updates to member information are reflected in inbox
     chat: ucase          Updates to member information are reflected in threads
     chat:   uri  tested  POST /chat/typing
     chat:   uri  notest  POST /chat/poll
     chat:   uri  tested  GET /chat/thread
     chat:   uri  tested  GET /chat
     chat:   uri  tested  POST /chat/send
     chat: evt->     sub  registration-approved
     chat: evt->     sub  member-blocked
     chat: evt->     sub  message-sent
     chat: evt->     sub  member-unblocked
     chat: evt->     sub  photo-approved
     chat: ->evt     pub  message-sent
     chat: ->evt     pub  member-typing
     chat: ->evt     pub  member-read-thread
```

Check out the code

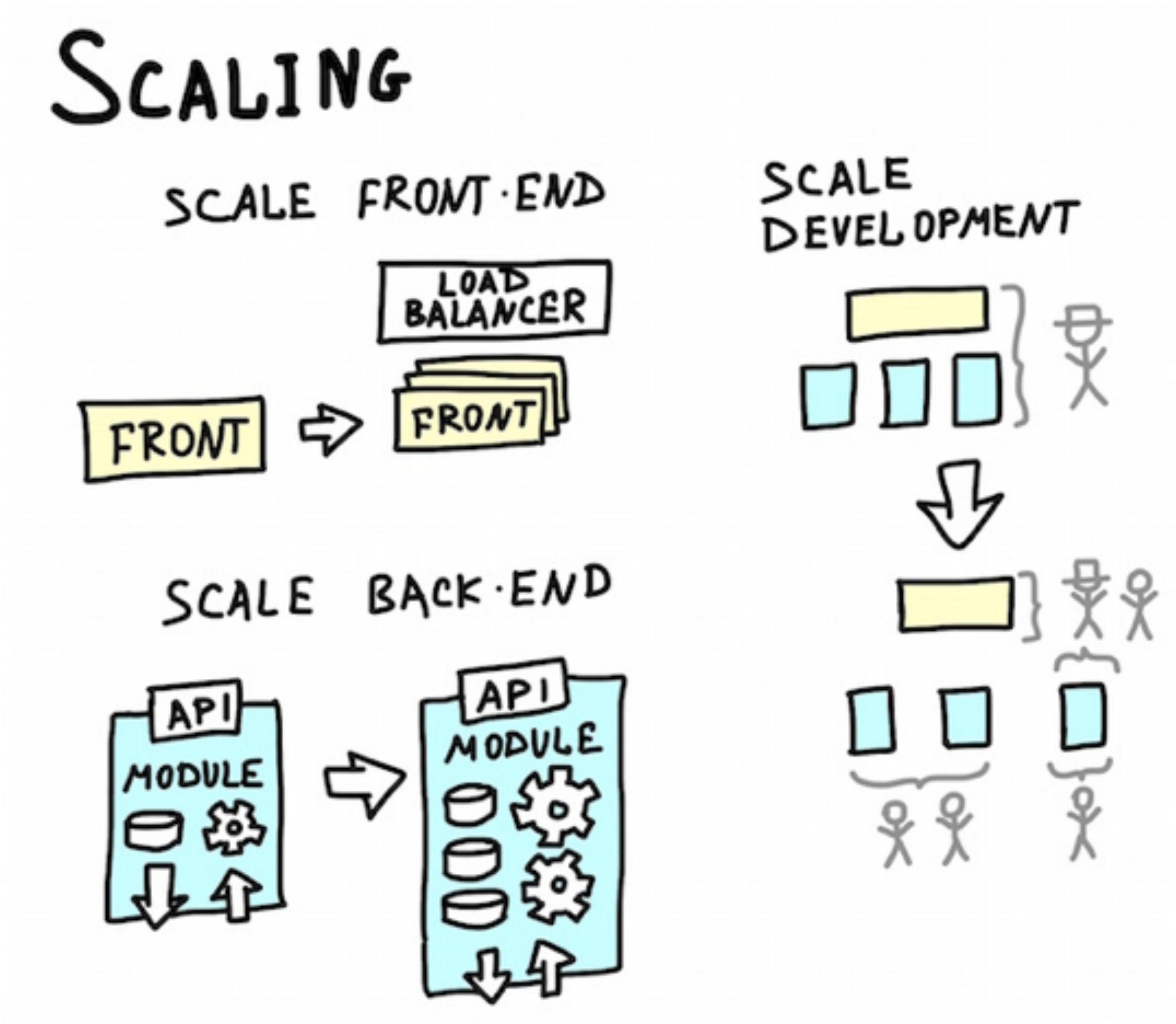
API Use Cases

Many paths of an emergent design. Apply to each component separately.



An example of evolution paths for a component. Each project will have a different version.

abdullin.com
@abdullin

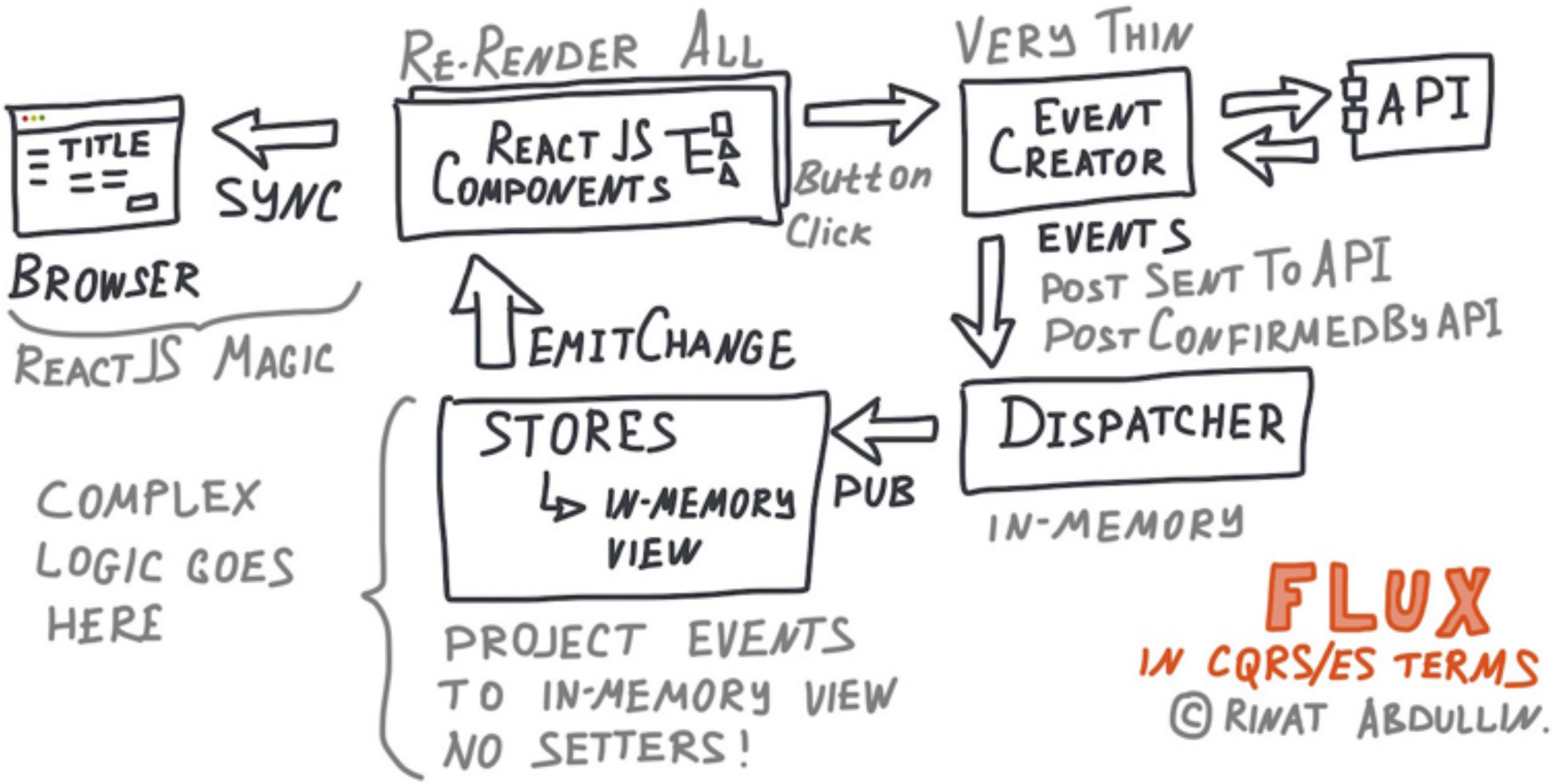


Divide and Conquer

To manage, optimize and scale individually

Front-end

Where the real fun starts



Flux/ReactJS

Event-driven, one of many options for UI/UX Design

Questions?

Time for CQRS Beers!