# µServices

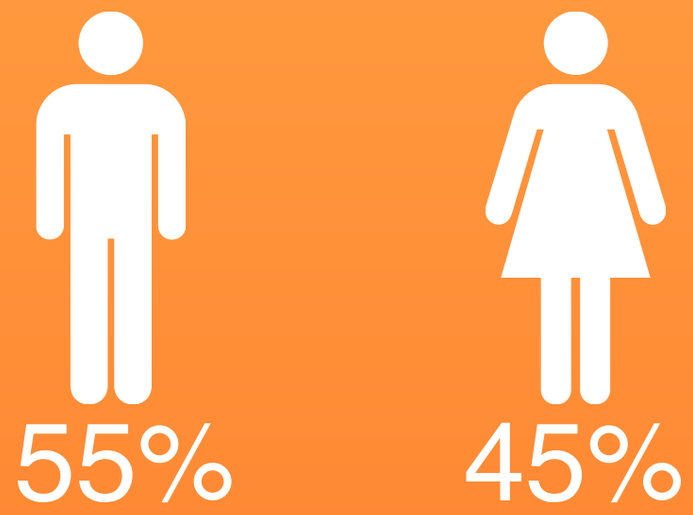

@pjvds


HAPPYPANCAKE.COM

# HAPPY PANCAKE™

28 000 000

page views per week

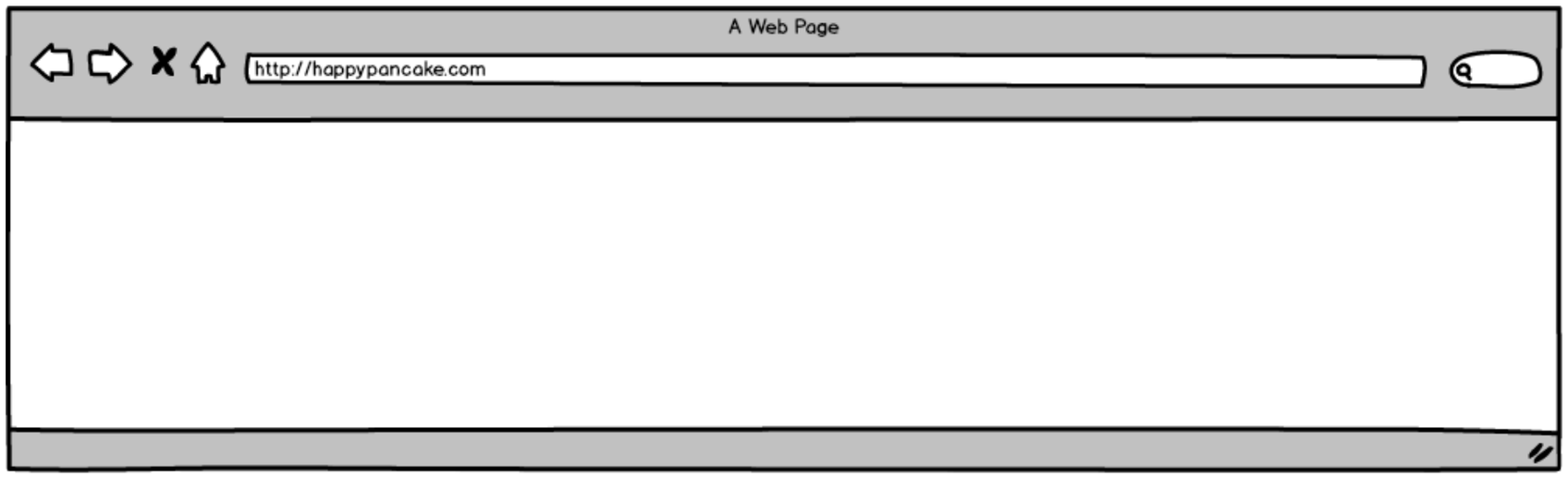



18 MINUTER

time spend per visit*

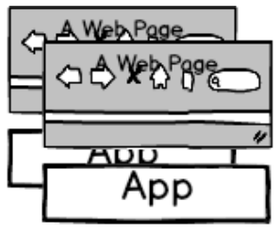
*Google Analytics v.17 2014

HAPPYPANCAKE™




App

Database




Database

A monolithic design is characterised by such a tight coupling among modules that they have no independent existence.

# challenges

- If one things fails, everything fails
- Can’t scale the database any further
- Expensive hardware
- Maintenance is scary
- Platinum support contract costs

# Goals

- Scaling individual parts of the system
- Lower hardware costs
- No platinum support
- Environment to quickly evaluate ideas


This is where
our story begins . . .

# Top inspirations

- Larry Constantine - Structured Design (1960)
- Douglas McIlroy - Programs (1965)
- Ken Thompson - Unix (1973)
- Antony Williams - COM (1987)
- Udi Dahan - Autonomous Components (2004)
- Greg young - CQRS (2008)
- Fred George - Micro Services (2013)

“Write simple parts connected by clean interfaces.”

“Design programs to be connected to other programs. “

–Ken Thompson, Unix Philosophy

“Make each program do one thing well. To do a new job, build afresh rather than complicate old programs by adding new features.”

“Expect the output of every program to become the input to another, as yet unknown, program.”

–Mc McIlroy, Unix Time-Sharing System Forward”. 1978

“A model can either be optimised for reading or for writing”

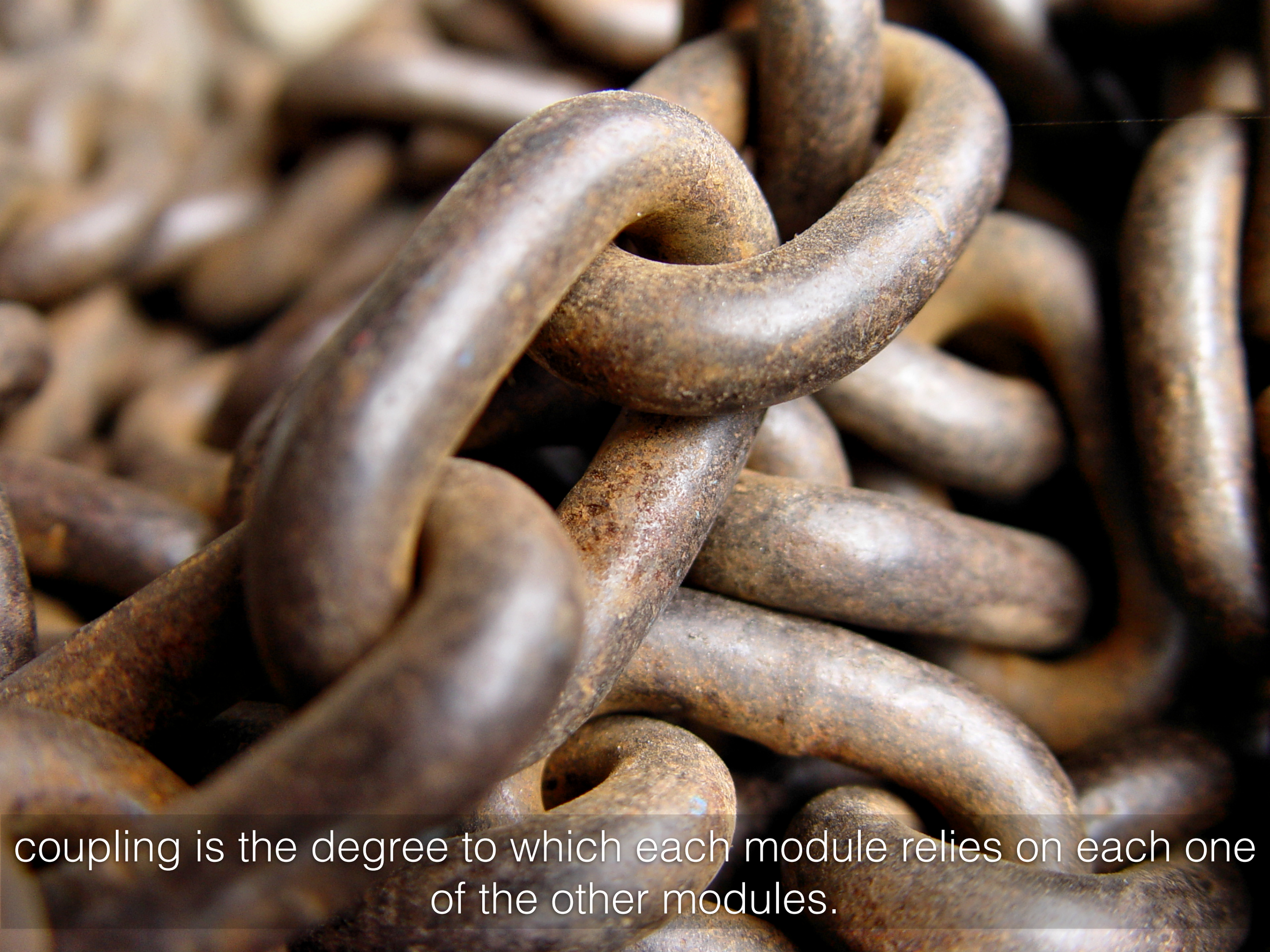
–Greg Young

“Depend on others and fail together”

–Rob Pike


simple: easily understood or done


autonomous: acting independently



coupling is the degree to which each module relies on each one of the other modules.

“µServices should be made as small as possible, but not smaller”

## INVITE THE RIGHT PEOPLE

- ☐ THOSE WHO BRING THE QUESTIONS
- ☐ THOSE WHO KNOW THE ANSWERS*)

*) AND DON'T WORRY IF YOU DON'T GET ALL THE ANSWERS.

## UNLIMITED MODELING SPACE

- CAN'T TELL A PROBLEM SIZE BEFORE EXPLORING

## FOCUS ON DOMAIN EVENTS

- SIMPLE SEMANTIC & NOTATION
- EASY TO GRASP BY EVERYBODY IN THE ROOM

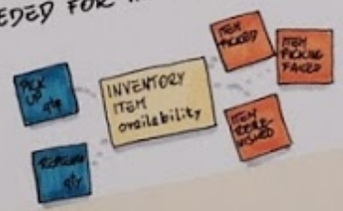

## WHAT DOES TRIGGER EVENTS?

- A COMMAND FROM A GIVEN USER
- OR AN EXTERNAL SYSTEM
- OR SIMPLY THE TIME END OF MONTH
- OR JUST ANOTHER EVENT ITEM ADDED TO CART PRICE UPDATED

## EXPLORE AGGREGATES OUTSIDE-IN

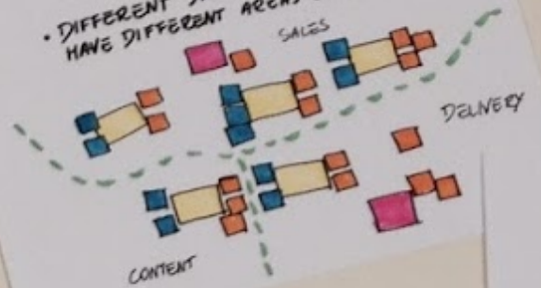
AN AGGREGATE CAN ACCEPT OR REJECT COMMANDS...

...AND MOSTLY NEEDS ONLY THE DATA NEEDED FOR THIS PURPOSE



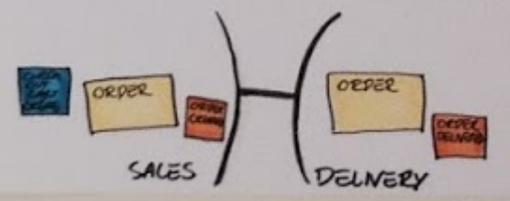

## EXPLORE SUBDOMAINS

- DIFFERENT STAKEHOLDERS WILL HAVE DIFFERENT AREAS OF EXPERTISE




## EXPLORE BOUNDED CONTEXTS

- WHEN DISCUSSIONS ARISE AROUND MEANING AND SEMANTICS, MULTIPLE MODELS ARE PROBABLY INVOLVED



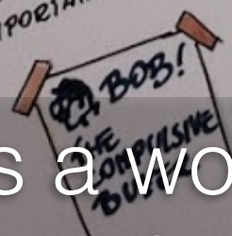

## SKETCH ACCEPTANCE TESTS

- CRITICAL OR AMBIGUOUS USE CASES MIGHT REQUIRE A PRECISE COMPLETION CRITERIA

GIVEN A REGISTERED CUSTOMER BOB
AND BOB IS A PREMIUM CUSTOMER
WHEN BOB PURCHASES GOODS FOR 100€
THEN BOB IS GIVEN A 20% DISCOUNT

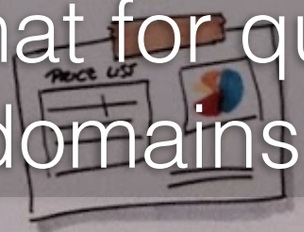
## PERSONAS & USER GOALS

- SOMETIMES WHO IS PERFORMING THE ACTION IS MORE IMPORTANT THAN OTHER DETAILS

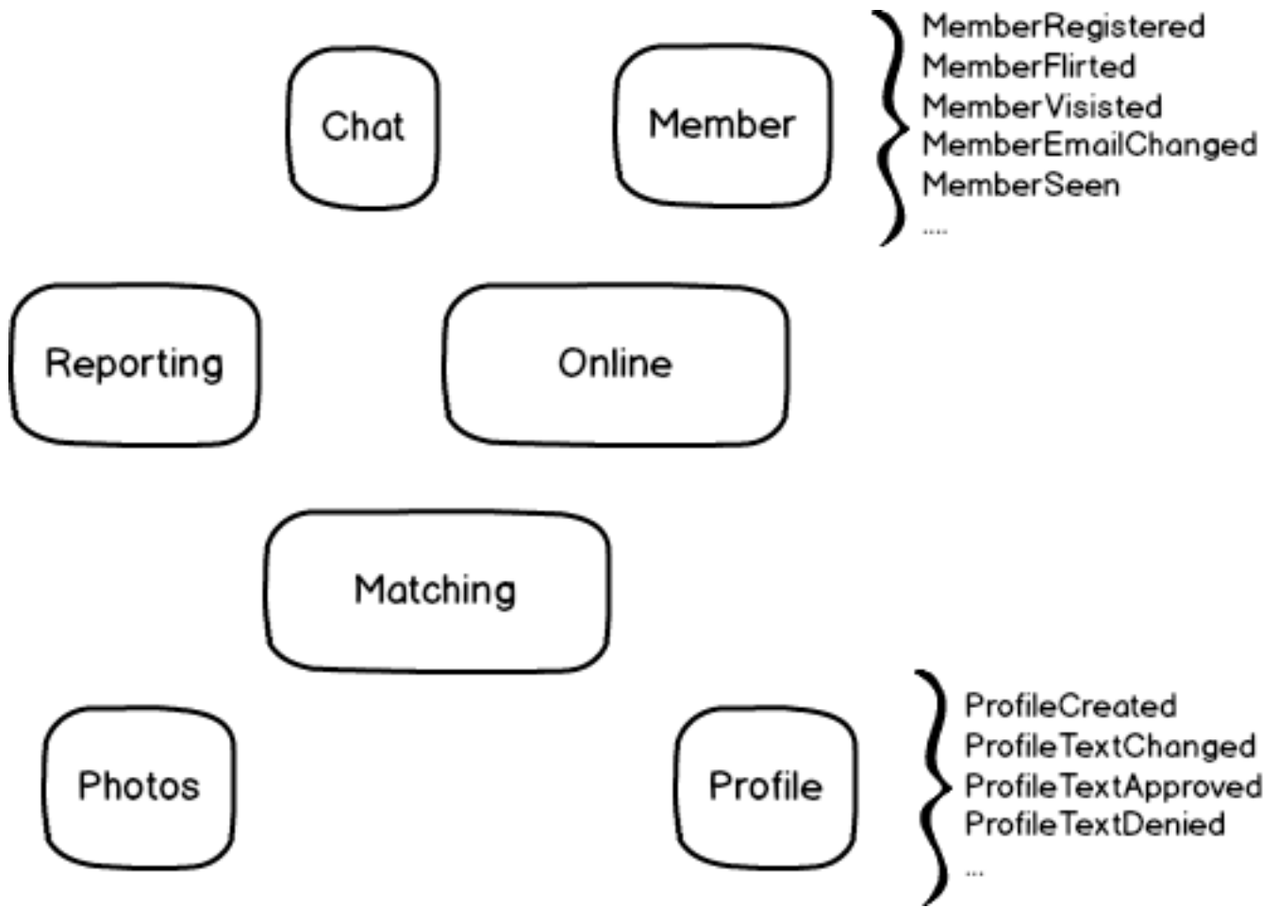



## SKETCHING READ MODEL ARTIFACTS

- SOME SCENARIOS REQUIRE A WELL DEFINED SET OF INFORMATION TO BE DISPLAYED TO THE USER

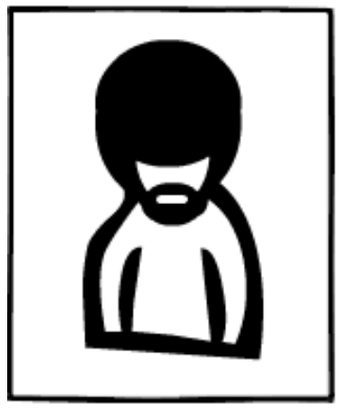



Event Storming is a workshop format for quickly exploring complex business domains.


Chat
Member
MemberRegistered
MemberFlirted
MemberVisisted
MemberEmailChanged
MemberSeen
....
Reporting
Online
Matching
Photos
Profile
ProfileCreated
ProfileTextChanged
ProfileTextApproved
ProfileTextDenied
...

# where does online status come from?

## Bob Ross

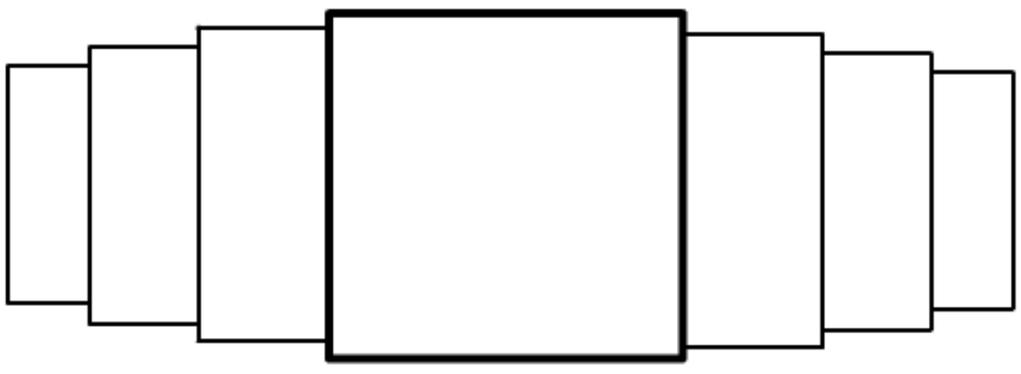


Bob Ross, born in Florida on October 29, 1942, discovered oil painting while he was enlisted in the U.S. Air Force in the early 1960s. He studied the "wet-on-wet" technique, which allowed him to produce complete paintings in less than an hour. He then became an instructor himself, eventually teaching a TV audience of millions on the PBS show The Joy of Painting.

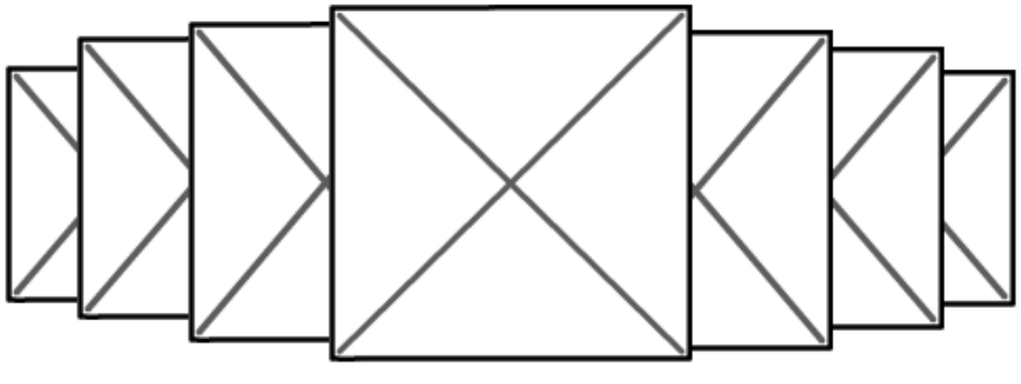
Flirt Chat

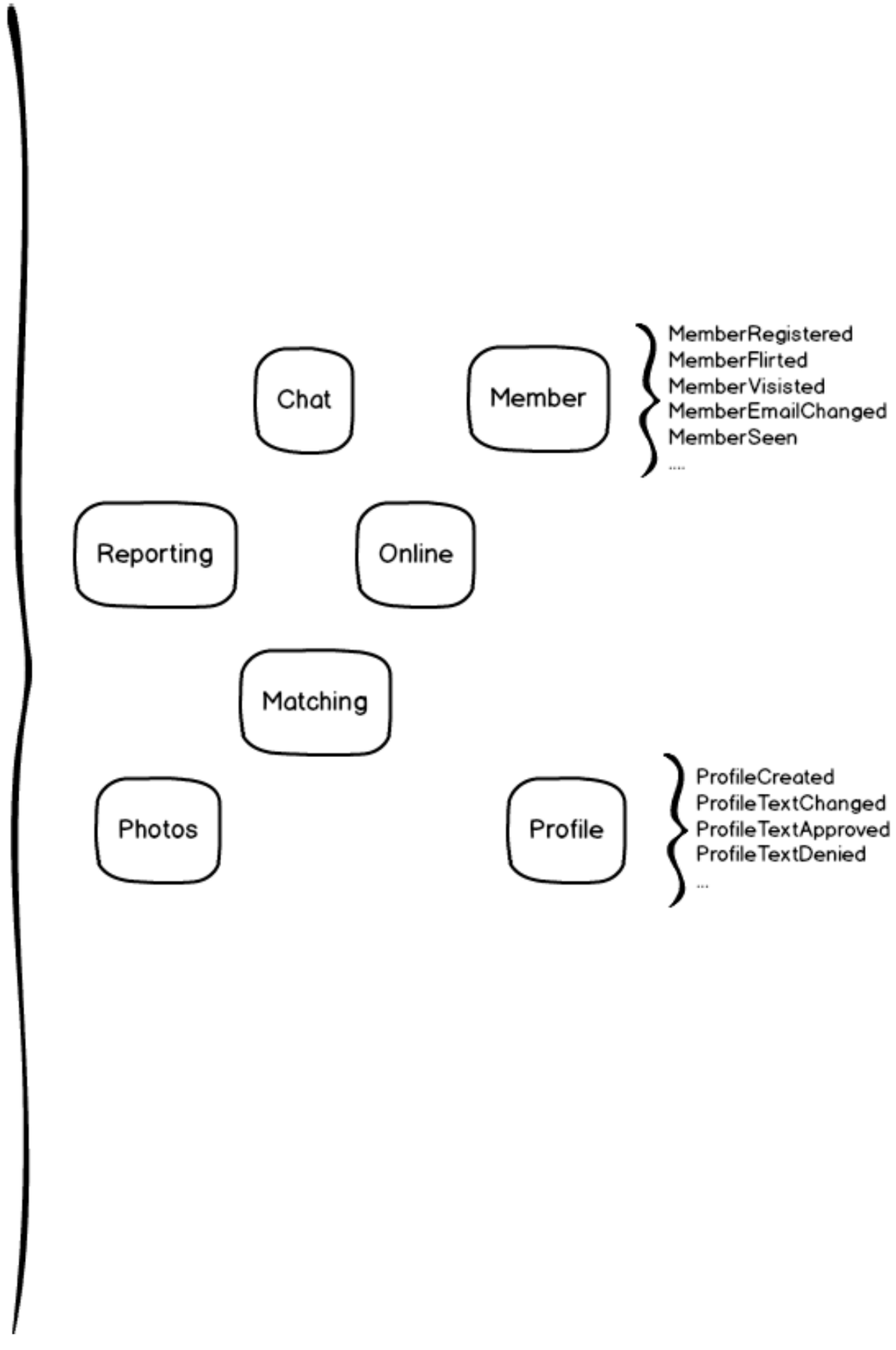
☑ Online Now!

Public Photo's



Private Photo's



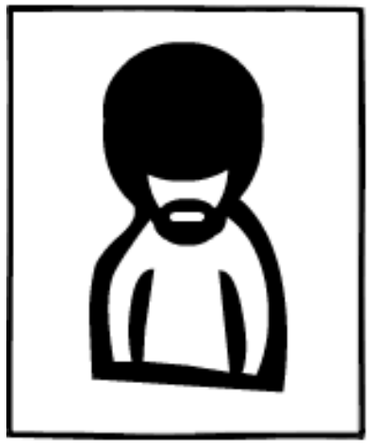


# STORIES FROM THE SOA

“No µService should ever depend on another being available”

“every view should be served by a single to one μService”

# Bob Ross

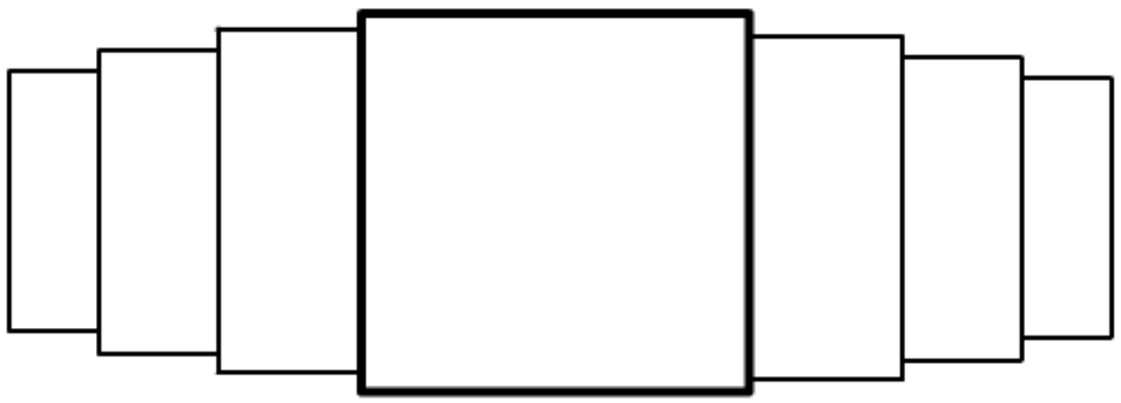


Bob Ross, born in Florida on October 29, 1942, discovered oil painting while he was enlisted in the U.S. Air Force in the early 1960s. He studied the "wet-on-wet" technique, which allowed him to produce complete paintings in less than an hour. He then became an instructor himself, eventually teaching a TV audience of millions on the PBS show The Joy of Painting.

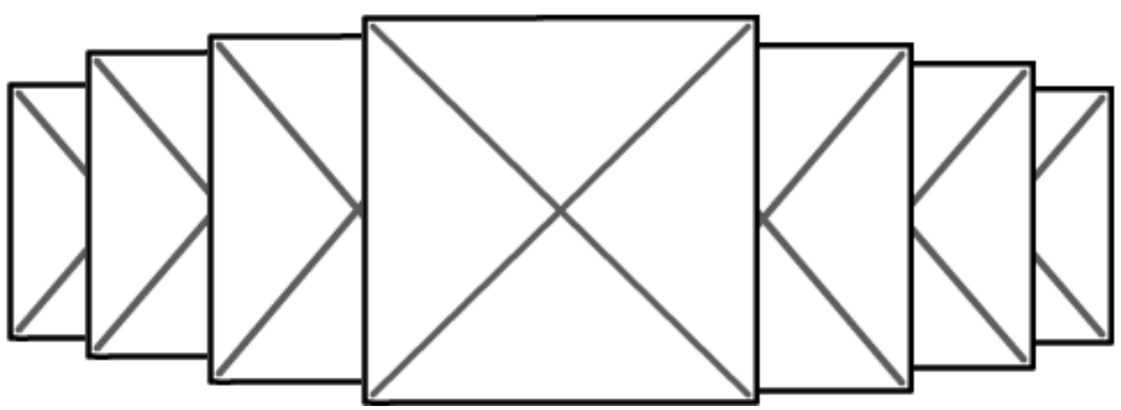
Flirt Chat

☑ Online Now!

Public Photo's

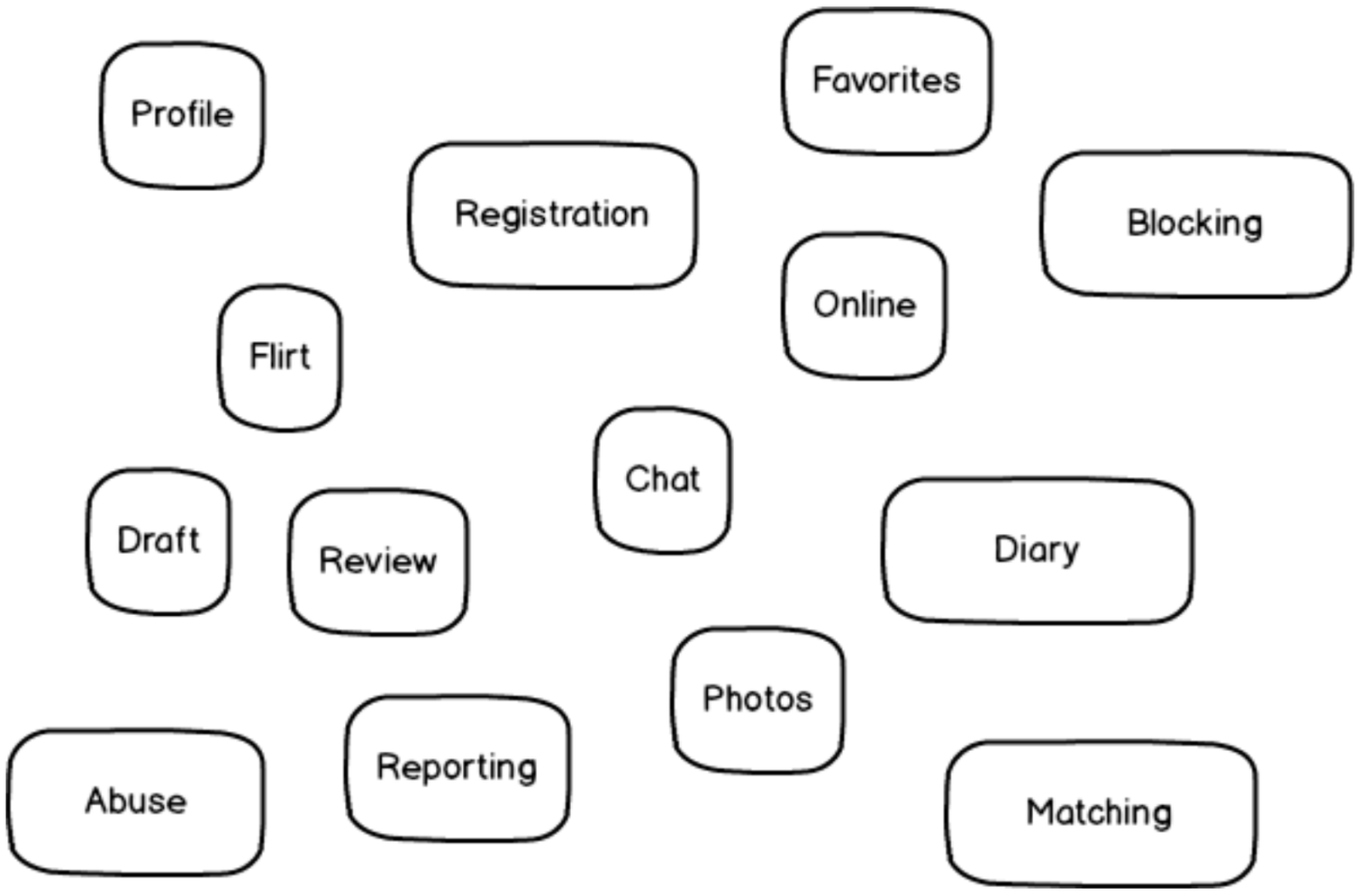


Private Photo's

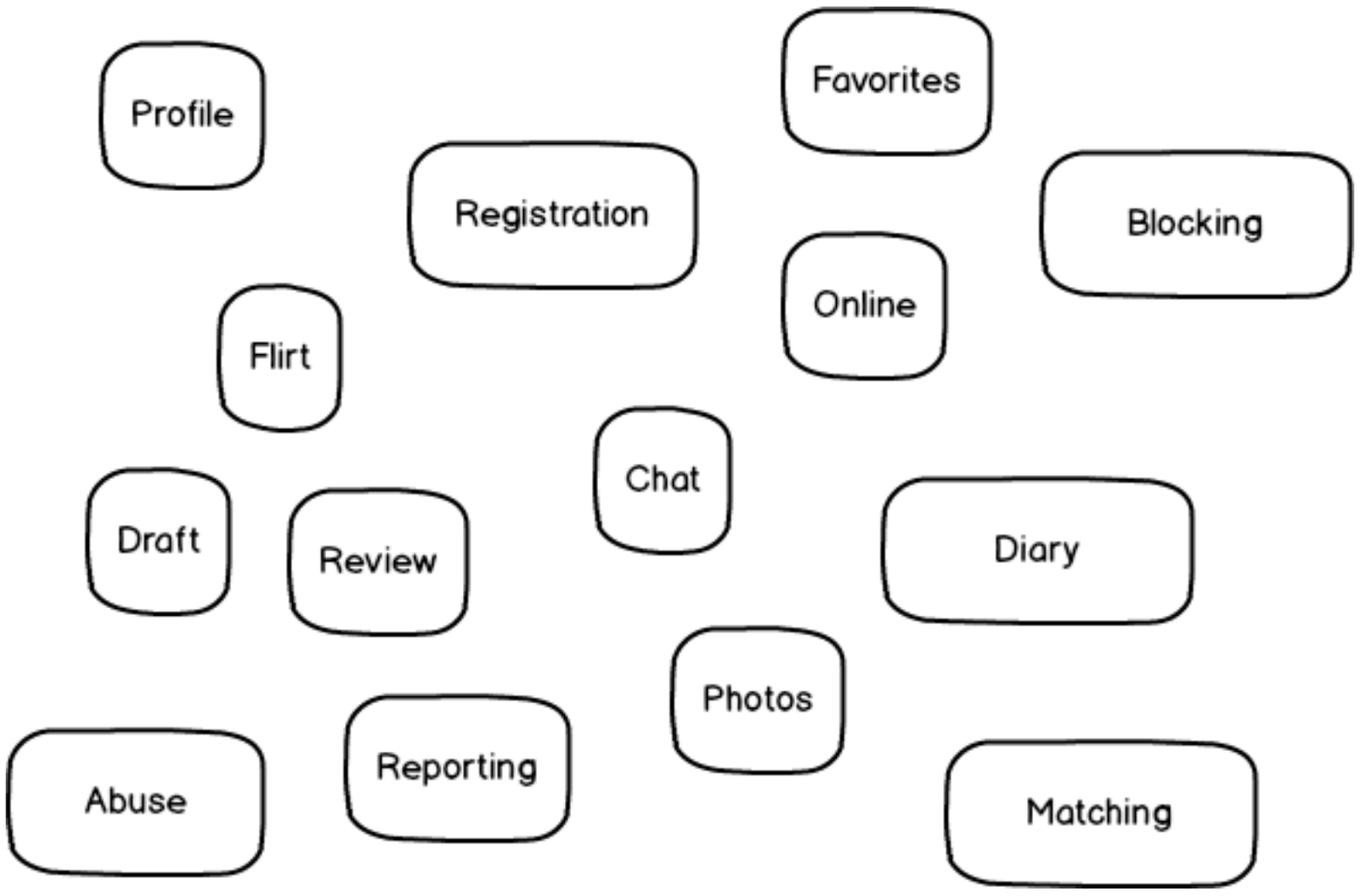


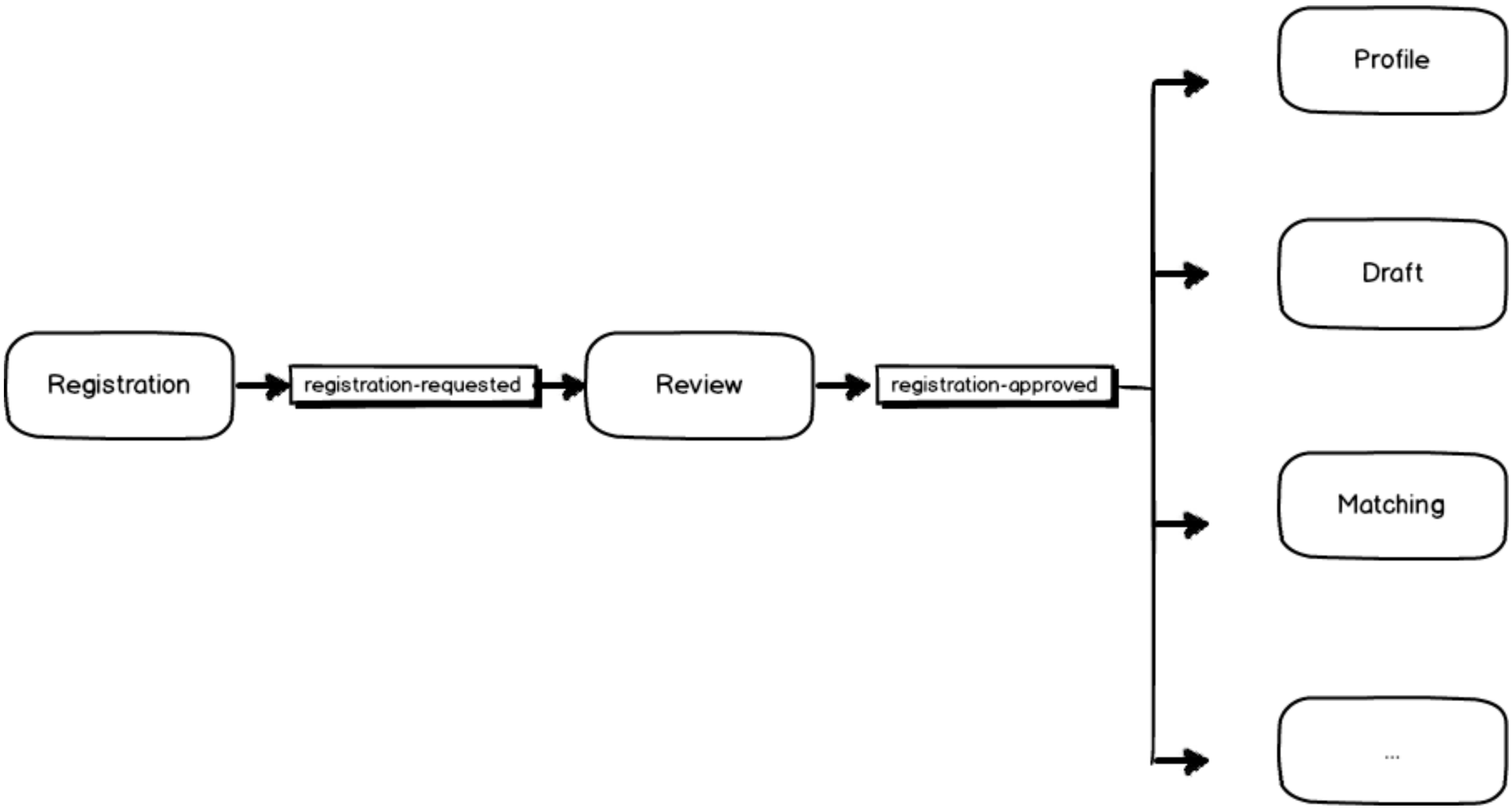
## GET /profile

```
{
   name: bob ross,
   text: …
   portrait: {
      url: /m/f.jpg,
      isLiked: true,
   }
   isOnline: true
   publicPhotos: …
   privatePhotos: [{
      url: /m/f.jpg
      hasAccess: true
   }]
}
```


Profile
Registration
Favorites
Blocking
Online
Flirt
Chat
Draft
Review
Diary
Photos
Reporting
Abuse
Matching

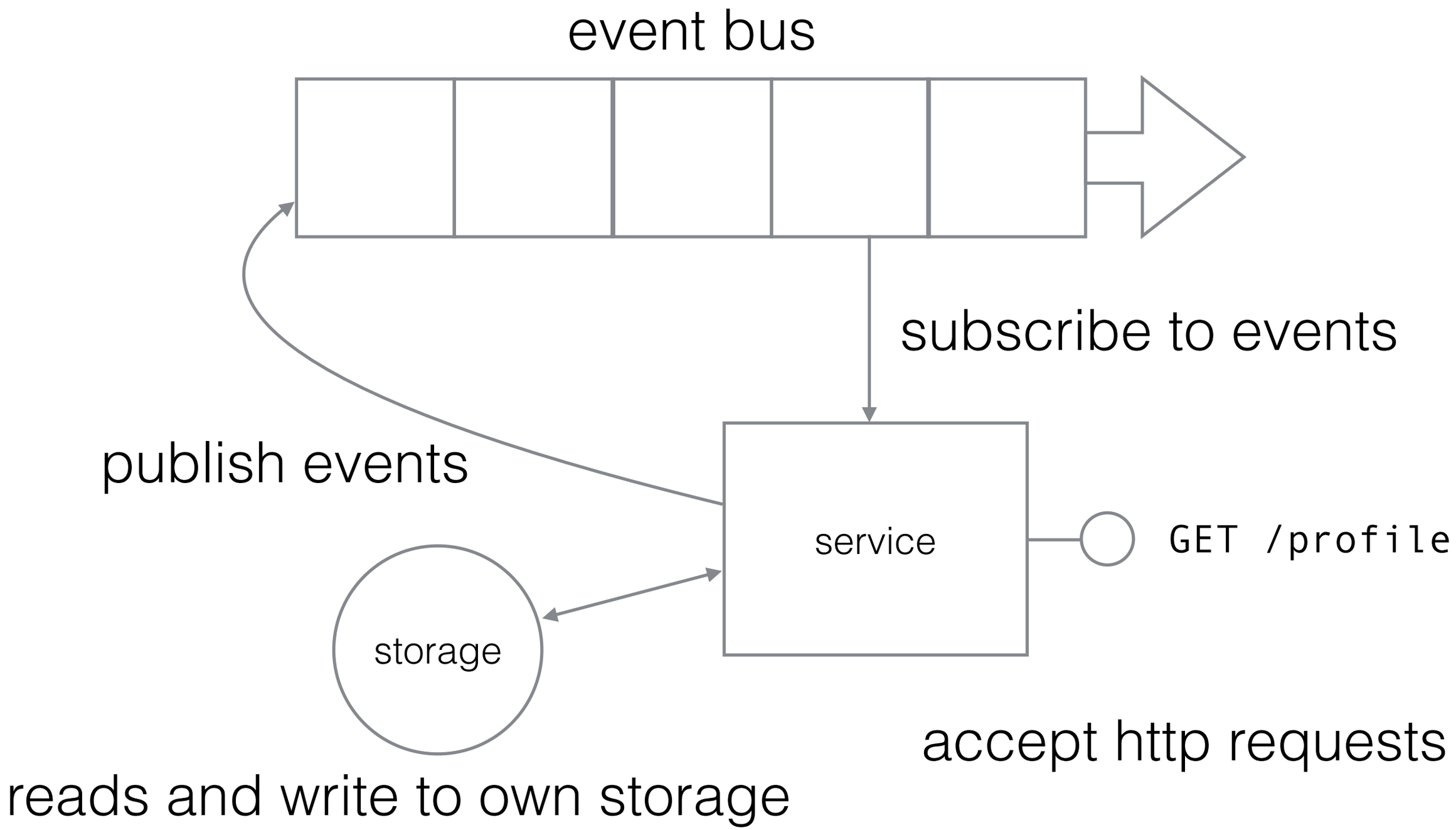
# where does the member live?




Registration
registration-requested
Review
registration-approved
Profile
Draft
Matching
...

“No µService represents an business entity, but rather serves an expect of one”

# µServices architecture



“An µService is the smallest consistency boundary”

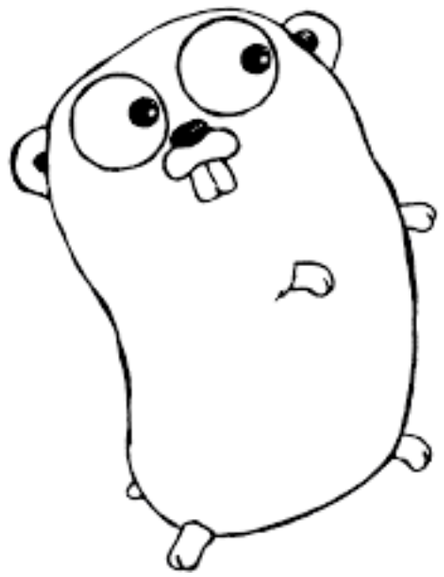
“µServices depends more on how the service fit into the environment than on how they are designed internally.”


docker


CoreOS


node
js


PostgreSQL




ERLANG


FOUNDATIONDB


Microsoft
Visual C#


mongoDB

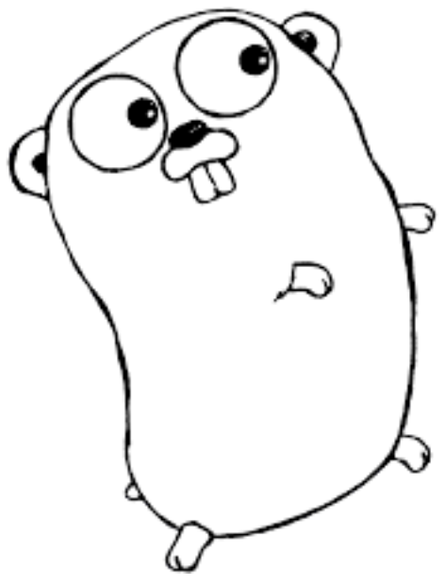
“Polyglot developers still hate context switching”

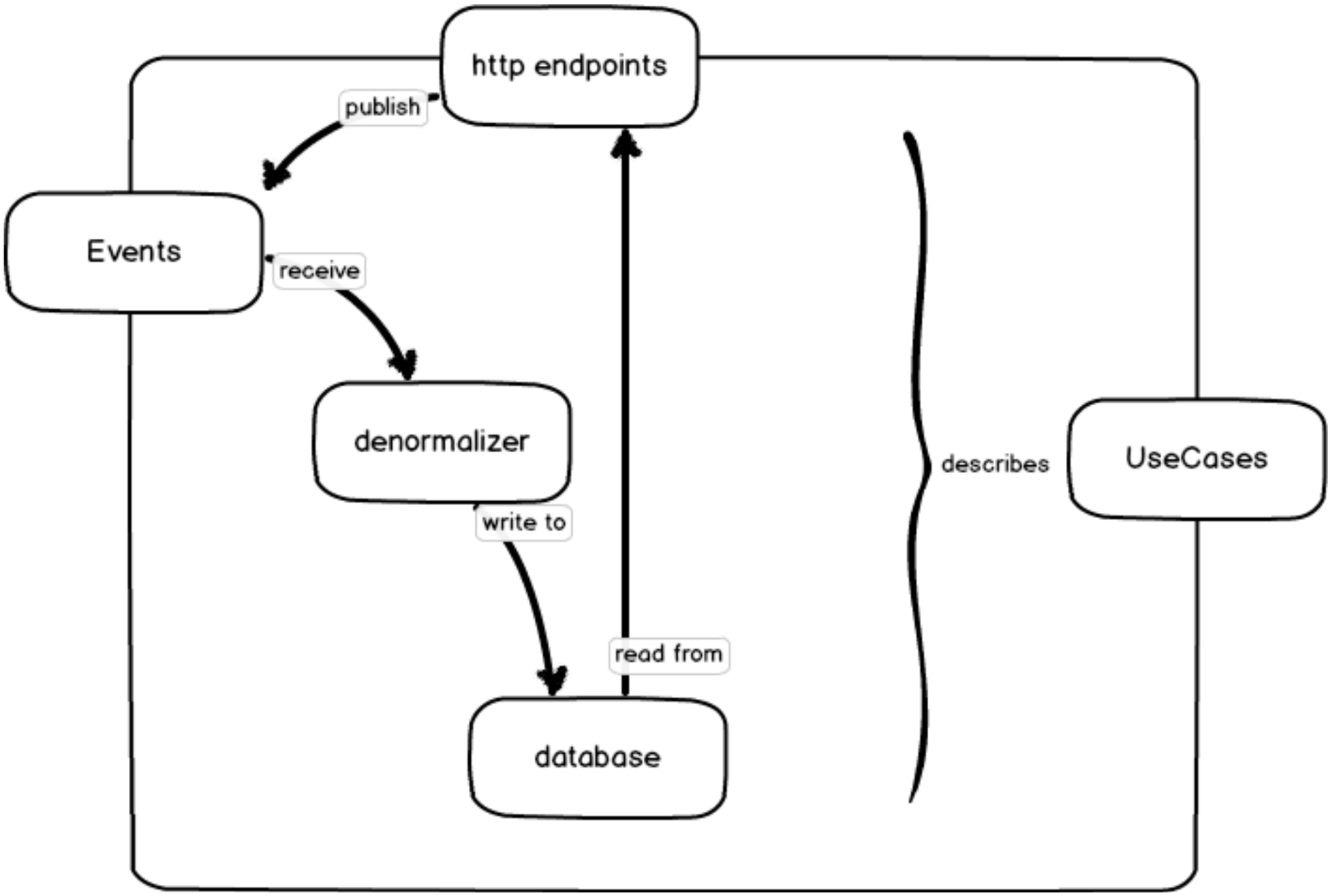
“Clarity is better than cleverness”


docker


PostgreSQL


node
js




http endpoints
publish
Events
receive
denormalizer
write to
read from
database
describes
UseCases

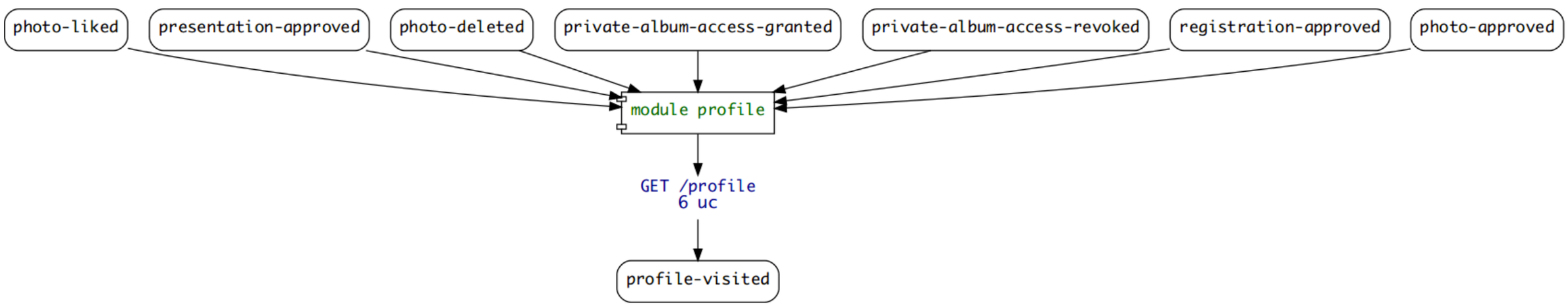
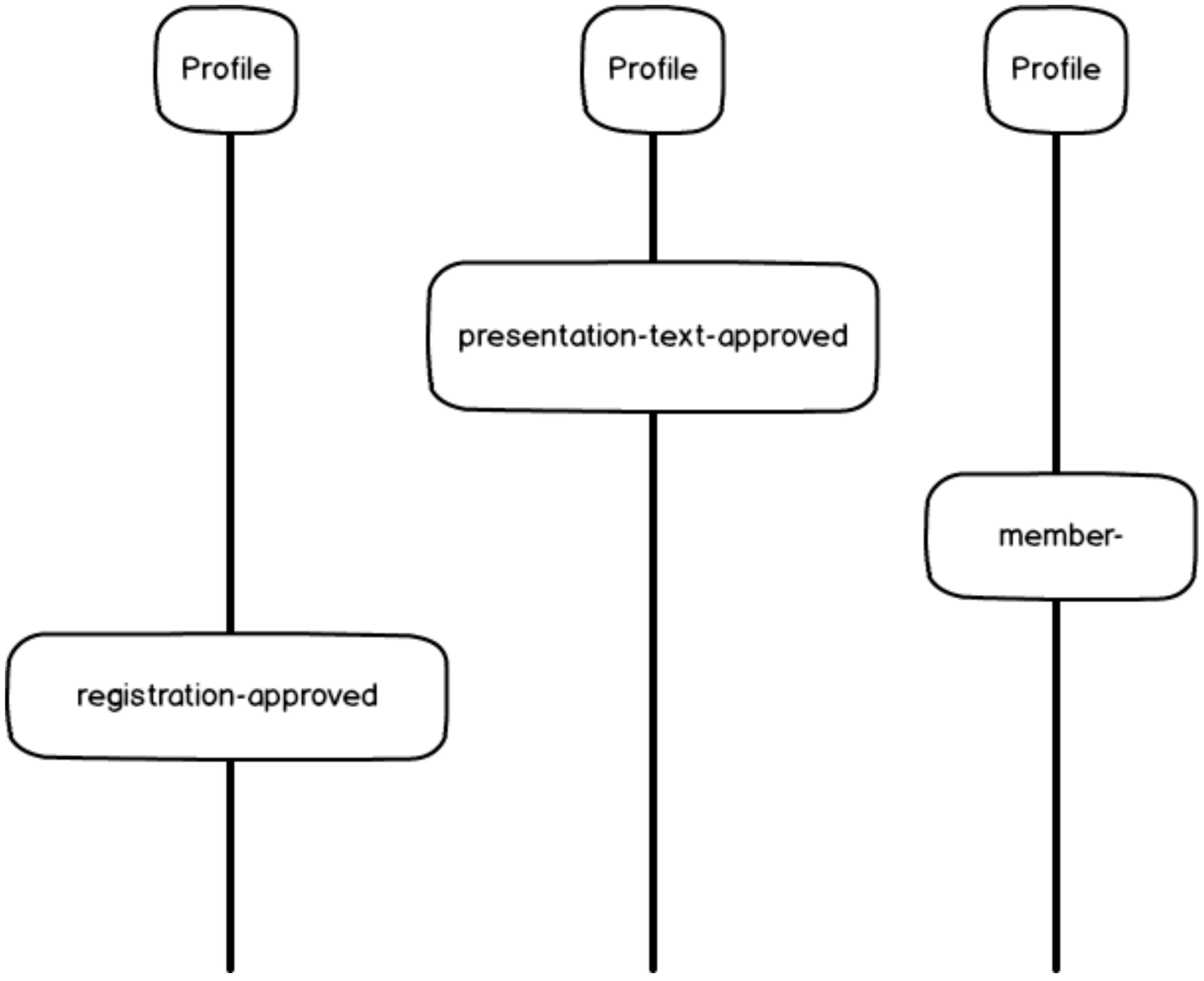
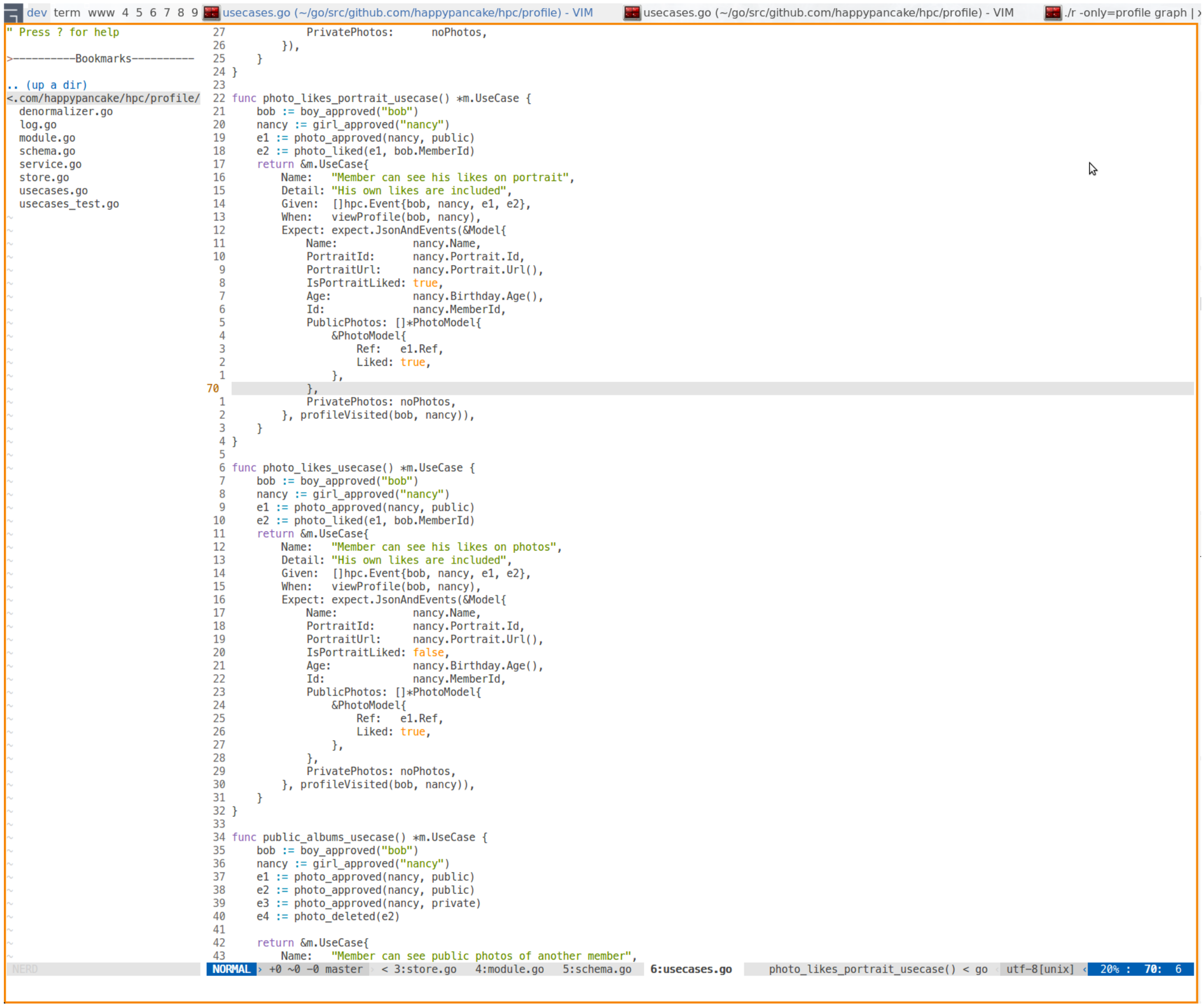
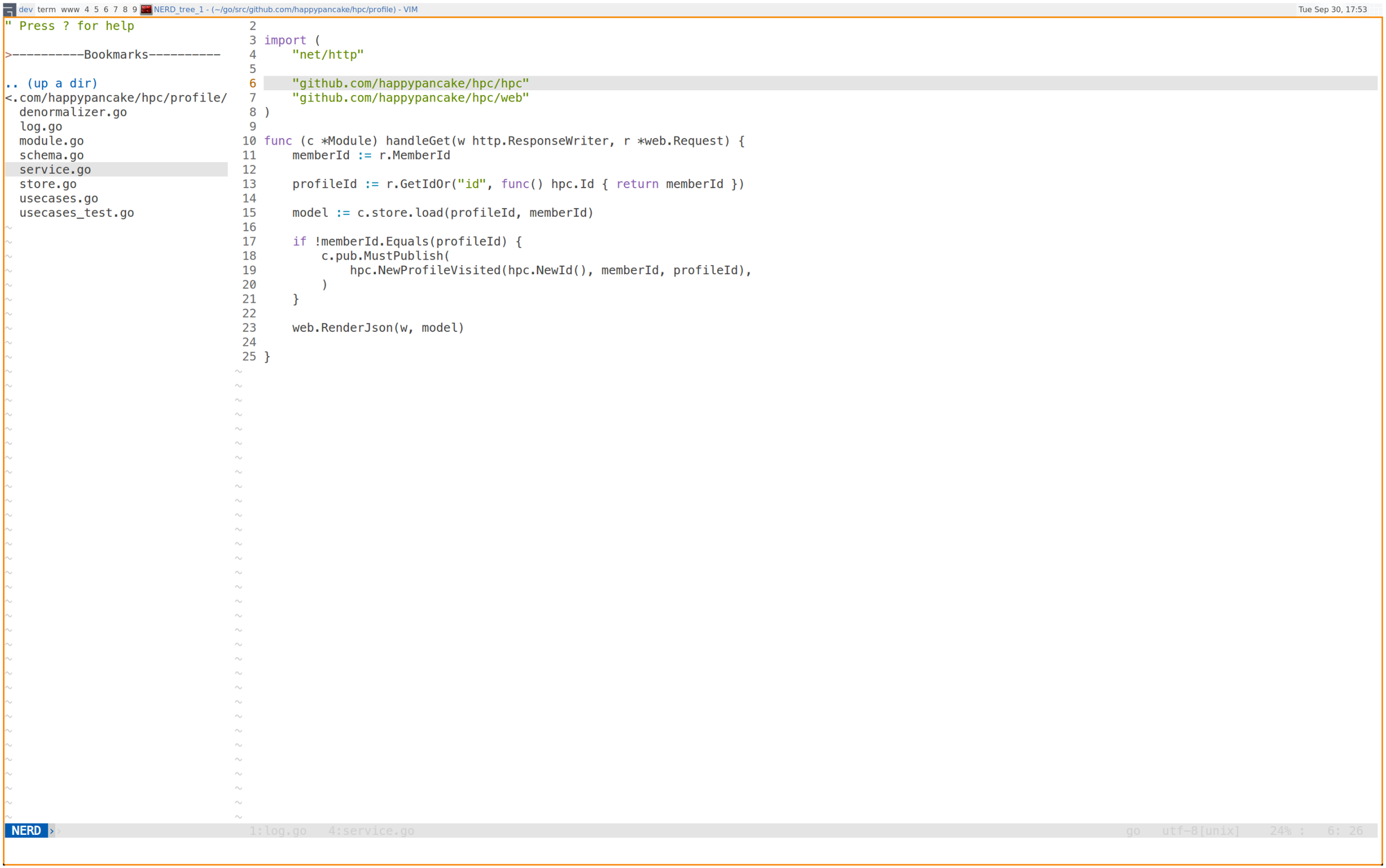


photo-liked
presentation-approved
photo-deleted
private-album-access-granted
private-album-access-revoked
registration-approved
photo-approved
module profile
GET /profile
6 uc
profile-visited

“An µService is the smallest part of the system that can scale independently”

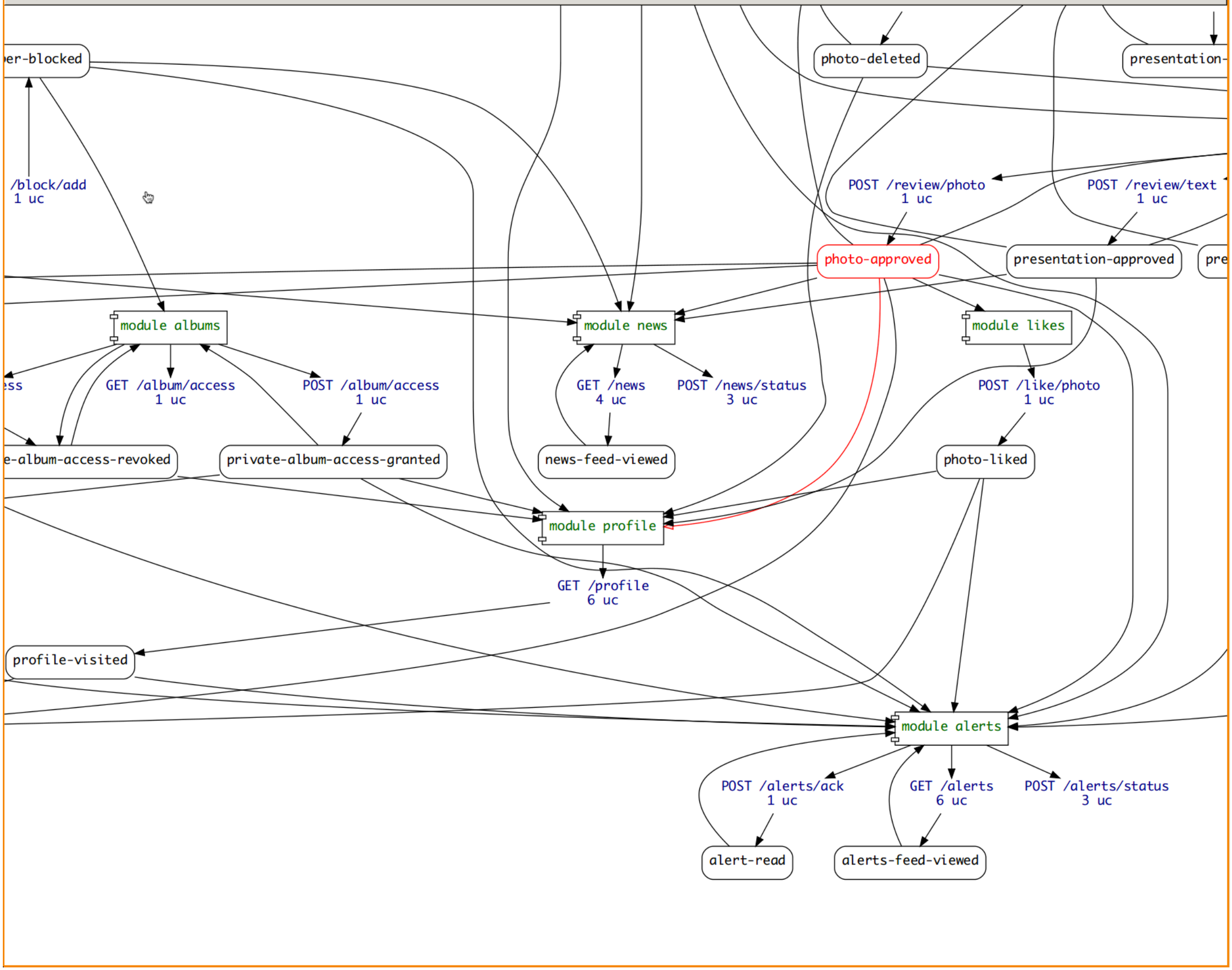

Profile
Profile
Profile
presentation-text-approved
registration-approved
member-


dev term www 4 5 6 7 8 9
usecases.go (~/go/src/github.com/happypancake/hpc/profile) - VIM
usecases.go (~/go/src/github.com/happypancake/hpc/profile) - VIM
./r -only=profile graph |
" Press ? for help
>----------Bookmarks----------
.. (up a dir)
<.com/happypancake/hpc/profile/
denormalizer.go
log.go
module.go
schema.go
service.go
store.go
usecases.go
usecases_test.go
PrivatePhotos: noPhotos,
}),
}
}
func photo_likes_portrait_usecase() *m.UseCase {
bob := boy_approved("bob")
nancy := girl_approved("nancy")
e1 := photo_approved(nancy, public)
e2 := photo_liked(e1, bob.MemberId)
return &m.UseCase{
Name: "Member can see his likes on portrait",
Detail: "His own likes are included",
Given: []hpc.Event{bob, nancy, e1, e2},
When: viewProfile(bob, nancy),
Expect: expect.JsonAndEvents(&Model{
Name: nancy.Name,
PortraitId: nancy.Portrait.Id,
PortraitUrl: nancy.Portrait.Url(),
IsPortraitLiked: true,
Age: nancy.Birthday.Age(),
Id: nancy.MemberId,
PublicPhotos: []*PhotoModel{
&PhotoModel{
Ref: e1.Ref,
Liked: true,
},
},
PrivatePhotos: noPhotos,
}, profileVisited(bob, nancy)),
}
}
func photo_likes_usecase() *m.UseCase {
bob := boy_approved("bob")
nancy := girl_approved("nancy")
e1 := photo_approved(nancy, public)
e2 := photo_liked(e1, bob.MemberId)
return &m.UseCase{
Name: "Member can see his likes on photos",
Detail: "His own likes are included",
Given: []hpc.Event{bob, nancy, e1, e2},
When: viewProfile(bob, nancy),
Expect: expect.JsonAndEvents(&Model{
Name: nancy.Name,
PortraitId: nancy.Portrait.Id,
PortraitUrl: nancy.Portrait.Url(),
IsPortraitLiked: false,
Age: nancy.Birthday.Age(),
Id: nancy.MemberId,
PublicPhotos: []*PhotoModel{
&PhotoModel{
Ref: e1.Ref,
Liked: true,
},
},
PrivatePhotos: noPhotos,
}, profileVisited(bob, nancy)),
}
}
func public_albums_usecase() *m.UseCase {
bob := boy_approved("bob")
nancy := girl_approved("nancy")
e1 := photo_approved(nancy, public)
e2 := photo_approved(nancy, public)
e3 := photo_approved(nancy, private)
e4 := photo_deleted(e2)
return &m.UseCase{
Name: "Member can see public photos of another member",
NERD
NORMAL +0 ~0 -0 master < 3:store.go 4:module.go 5:schema.go 6:usecases.go photo_likes_portrait_usecase() < go utf-8[unix] 20% : 70: 6


dev term www 4 5 6 7 8 9 NERD_tree_1 - (~/go/src/github.com/happypancake/hpc/profile) - VIM
Tue Sep 30, 17:53
" Press ? for help
>----------Bookmarks----------
.. (up a dir)
<.com/happypancake/hpc/profile/
denormalizer.go
log.go
module.go
schema.go
service.go
store.go
usecases.go
usecases_test.go
import (
"net/http"
"github.com/happypancake/hpc/hpc"
"github.com/happypancake/hpc/web"
)
func (c *Module) handleGet(w http.ResponseWriter, r *web.Request) {
memberId := r.MemberId
profileId := r.GetIdOr("id", func() hpc.Id { return memberId })
model := c.store.load(profileId, memberId)
if !memberId.Equals(profileId) {
c.pub.MustPublish(
hpc.NewProfileVisited(hpc.NewId(), memberId, profileId),
)
}
web.RenderJson(w, model)
}
NERD
1:log.go 4:service.go
go utf-8[unix] 24% : 6: 26

dev term www 4 5 6 7 8 9 Dot Viewer ./r graph | xdot usecases.go (~/go/src/githu

Open Refresh Print Zoom In Zoom Out Best Fit Normal Size

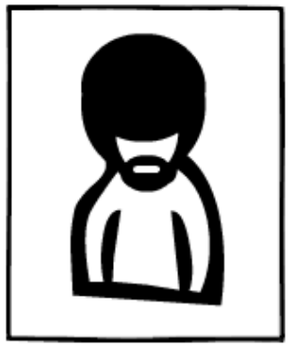


```
➜  hpc git:(master) ./r summary | grep profile:
  profile: ucase          Request without id returns profile of current user
  profile: ucase          Viewing profile shows info and publishes an event
  profile: ucase          Member can see public photos of another member
  profile: ucase          Member can see his likes on photos
  profile: ucase          Member with access can see private photos
  profile: ucase          Revoking access hides private photos
  profile:   uri  tested  GET /profile
  profile: evt->  tested  presentation-approved
  profile: evt->  tested  photo-deleted
  profile: evt->  tested  private-album-access-granted
  profile: evt->  tested  private-album-access-revoked
  profile: evt->  tested  registration-approved
  profile: evt->  tested  photo-approved
  profile: evt->  tested  photo-liked
  profile: ->evt  tested  profile-visited
```

Bob Ross

http://happypancake.com/profile/user1

# Bob Ross

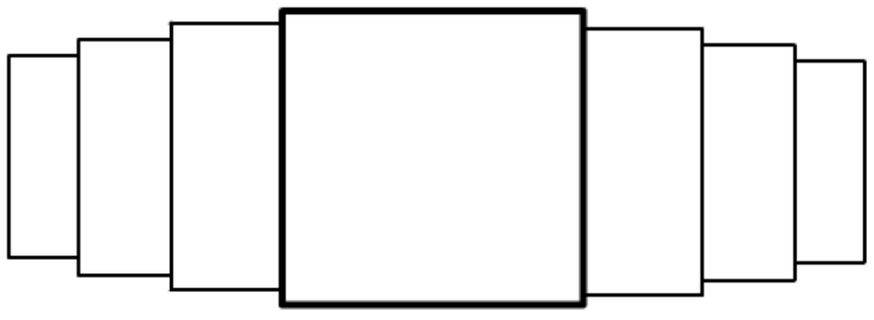


Bob Ross, born in Florida on October 29, 1942, discovered oil painting while he was enlisted in the U.S. Air Force in the early 1960s. He studied the "wet-on-wet" technique, which allowed him to produce complete paintings in less than an hour. He then became an instructor himself, eventually teaching a TV audience of millions on the PBS show The Joy of Painting.

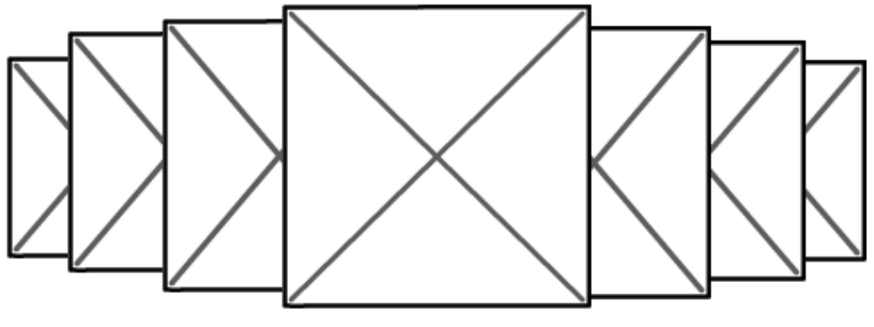
Flirt Chat

☑ Online Now!

## Public Photo's



## Private Photo's



## Online

Bob Ross

http://happypancake.com/profile/user1

```
<html>
  <head>
    ...
  </head>
  <body>
    <div place-holder-menu></div>
    <div place-holder-profile></div>
    <div place-holder-online></div>
  </body>
</html>
```

Bob Ross

http://happypancake.com/profile/user1

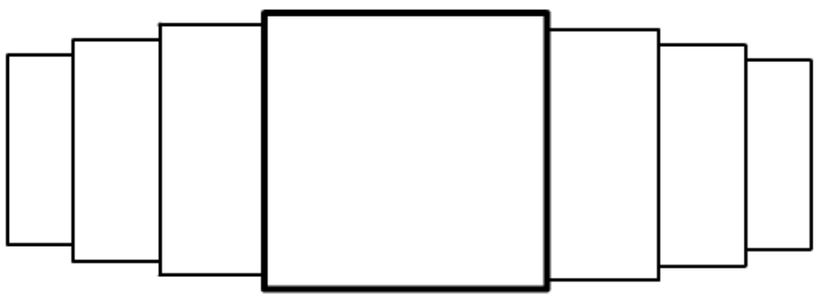
# Bob Ross

Bob Ross, born in Florida on October 29, 1942, discovered oil painting while he was enlisted in the U.S. Air Force in the early 1960s. He studied the "wet-on-wet" technique, which allowed him to produce complete paintings in less than an hour. He then became an instructor himself, eventually teaching a TV audience of millions on the PBS show The Joy of Painting.

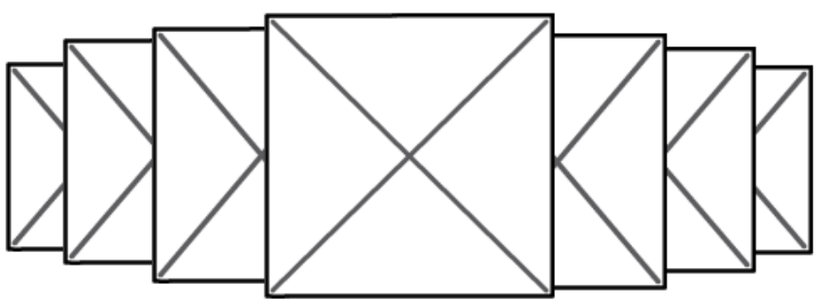
Flirt Chat

☑ Online Now!

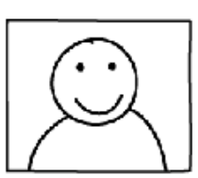
## Public Photo's



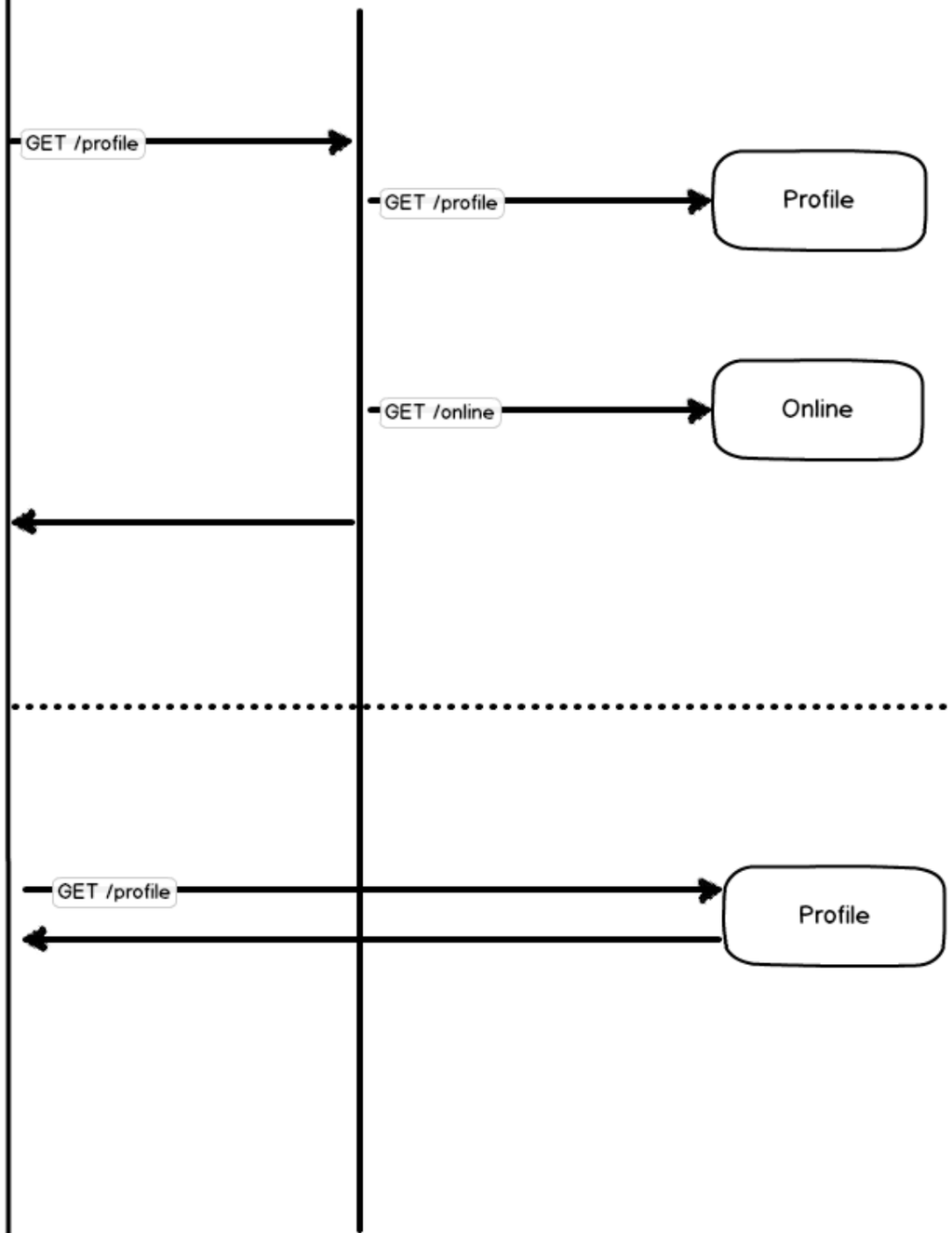
## Private Photo's

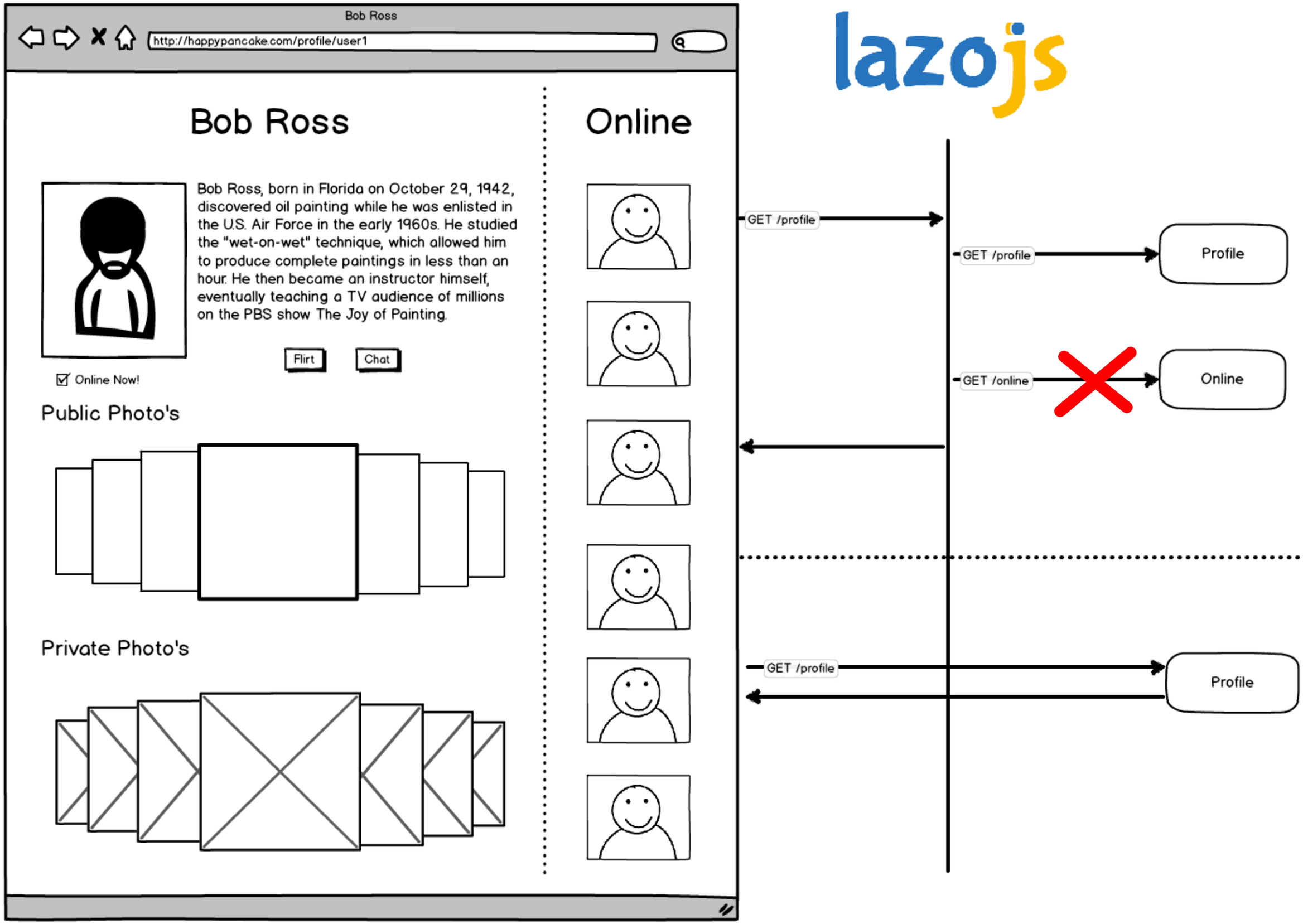


## Online



lazojs




Bob Ross
http://happypancake.com/profile/user1
Bob Ross
Bob Ross, born in Florida on October 29, 1942, discovered oil painting while he was enlisted in the U.S. Air Force in the early 1960s. He studied the "wet-on-wet" technique, which allowed him to produce complete paintings in less than an hour. He then became an instructor himself, eventually teaching a TV audience of millions on the PBS show The Joy of Painting.
Flirt
Chat
Online Now!
Public Photo's
Private Photo's
Online
lazojs
GET /profile
GET /profile
Profile
GET /online
Online
GET /profile
Profile

Bob Ross

http://happypancake.com/profile/user1

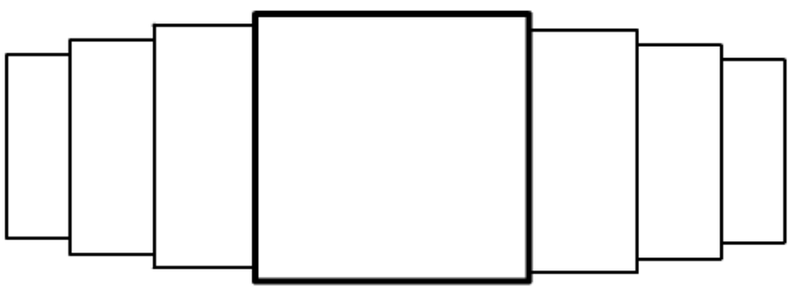
# Bob Ross

Bob Ross, born in Florida on October 29, 1942, discovered oil painting while he was enlisted in the U.S. Air Force in the early 1960s. He studied the "wet-on-wet" technique, which allowed him to produce complete paintings in less than an hour. He then became an instructor himself, eventually teaching a TV audience of millions on the PBS show The Joy of Painting.

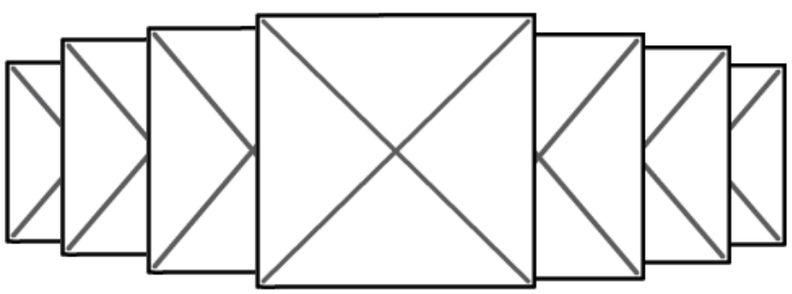
Flirt Chat

☑ Online Now!

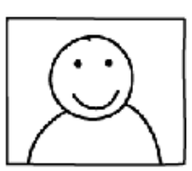
## Public Photo's



## Private Photo's

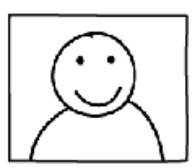


## Online



1: matches you

## Online

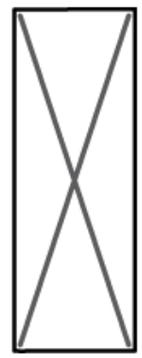


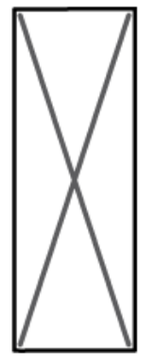
2: random match

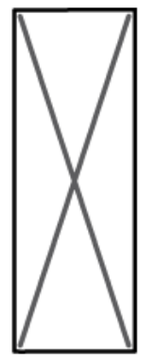
## Online

3: ads

## Ads







Bob Ross

http://happypancake.com/profile/user1

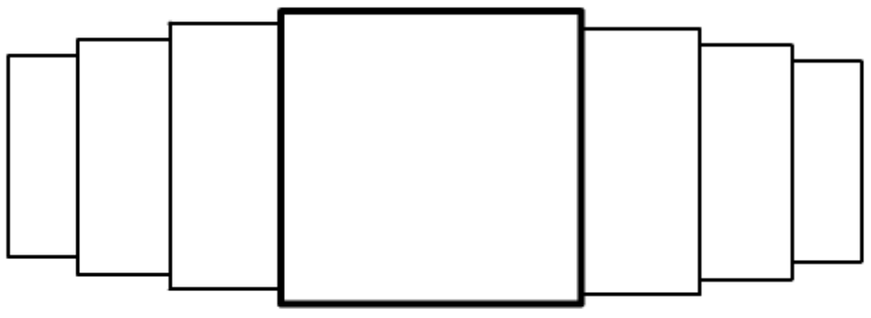
# Bob Ross

Bob Ross, born in Florida on October 29, 1942, discovered oil painting while he was enlisted in the U.S. Air Force in the early 1960s. He studied the "wet-on-wet" technique, which allowed him to produce complete paintings in less than an hour. He then became an instructor himself, eventually teaching a TV audience of millions on the PBS show The Joy of Painting.

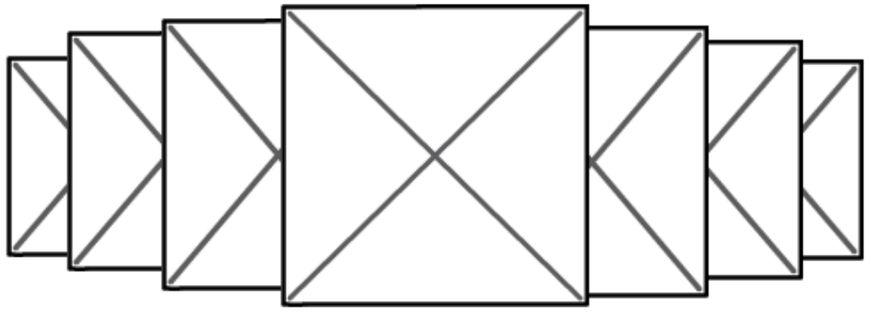
☐ Offline

Flirt Chat

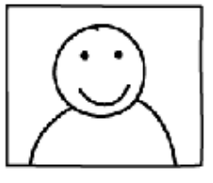
## Public Photo's

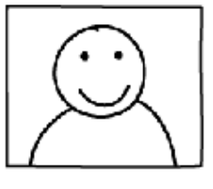


## Private Photo's



## Online





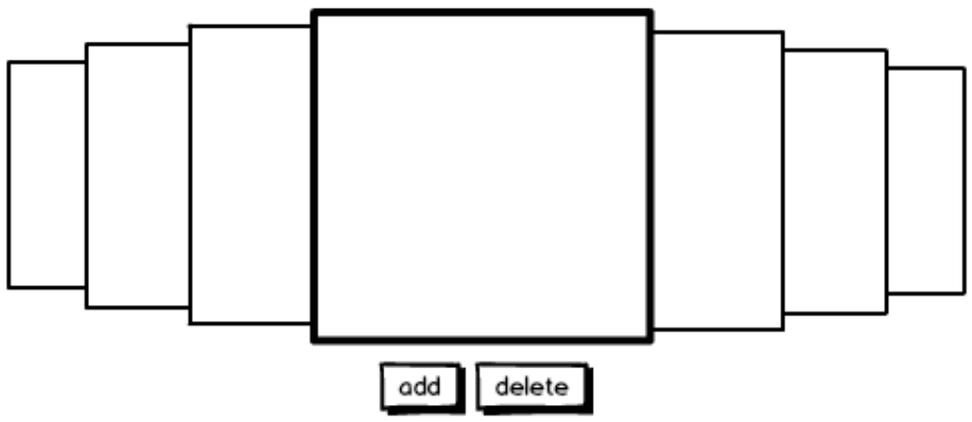
# Edit your profile

I like to paint

change

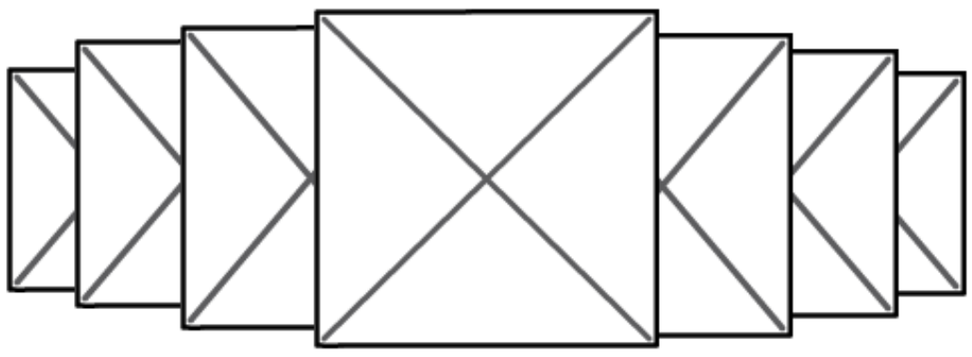
save

Public Photo's



add delete

Private Photo's



add delete

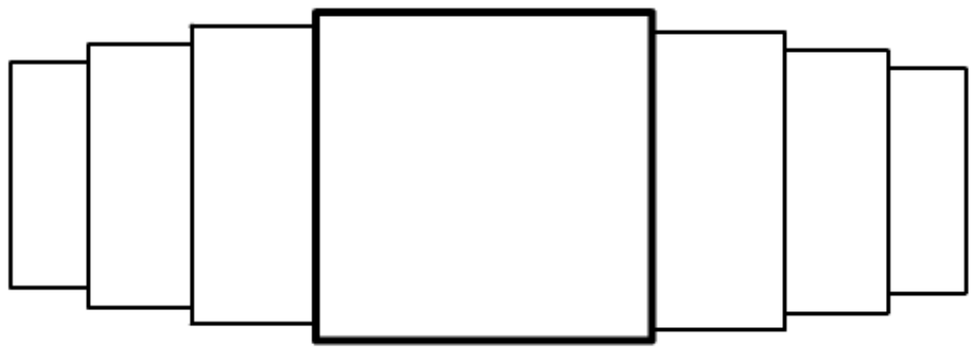
# Bob Ross

Bob Ross, born in Florida on October 29, 1942, discovered oil painting while he was enlisted in the U.S. Air Force in the early 1960s. He studied the "wet-on-wet" technique, which allowed him to produce complete paintings in less than an hour. He then became an instructor himself, eventually teaching a TV audience of millions on the PBS show The Joy of Painting.

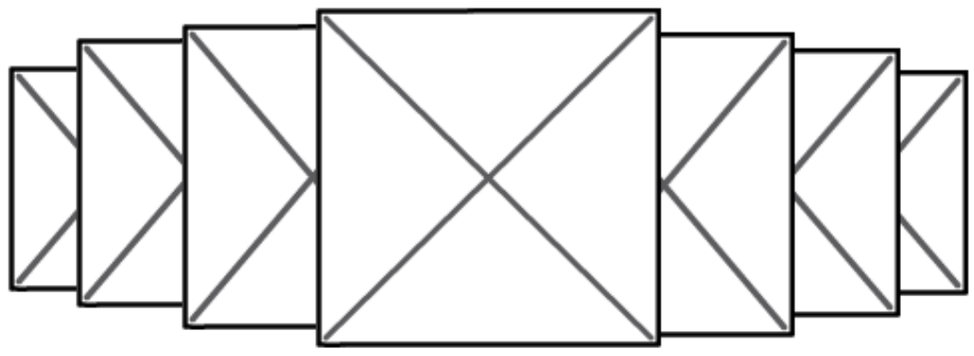
Flirt Chat

☑ Online Now!

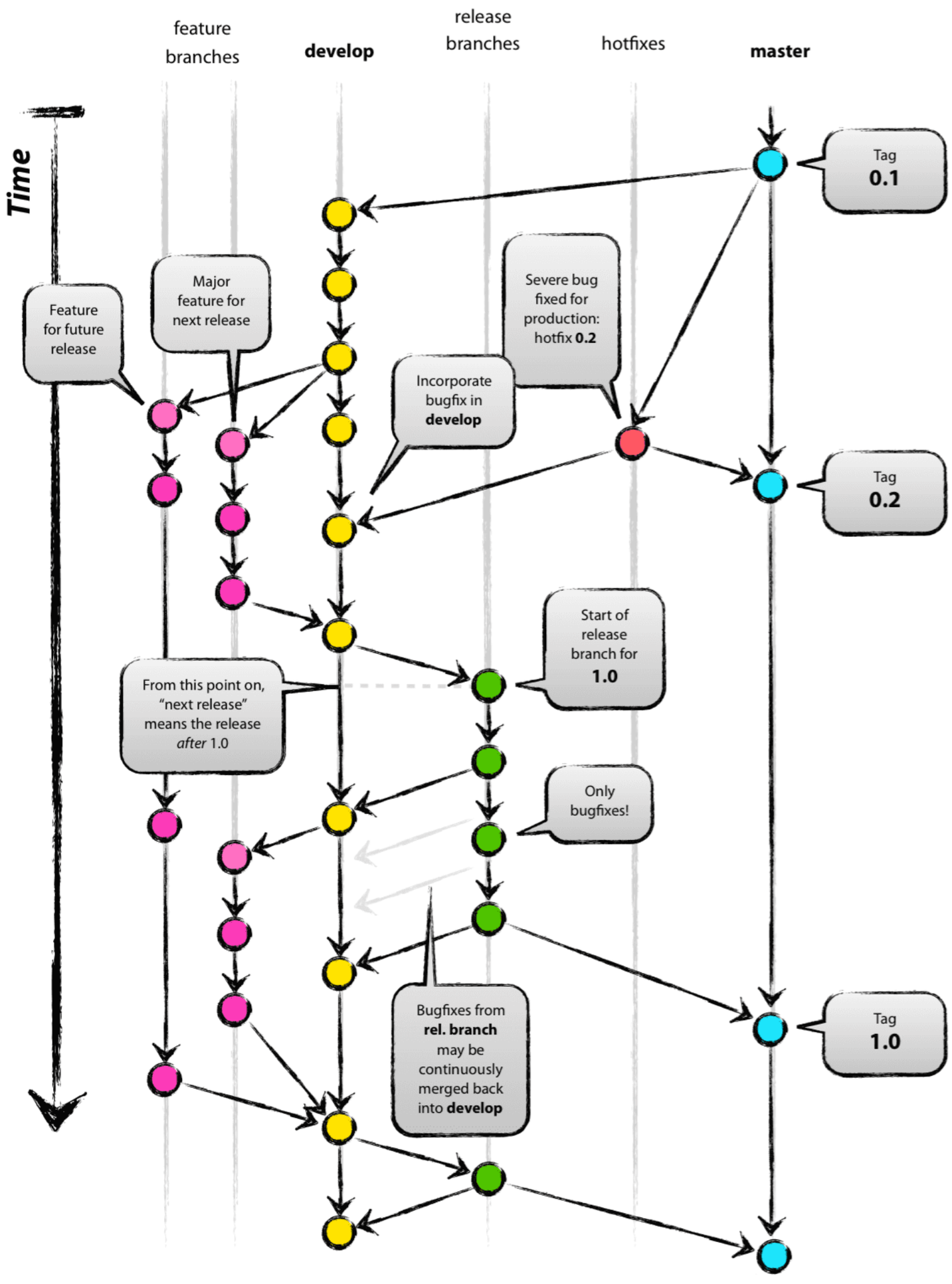
Public Photo's



Private Photo's

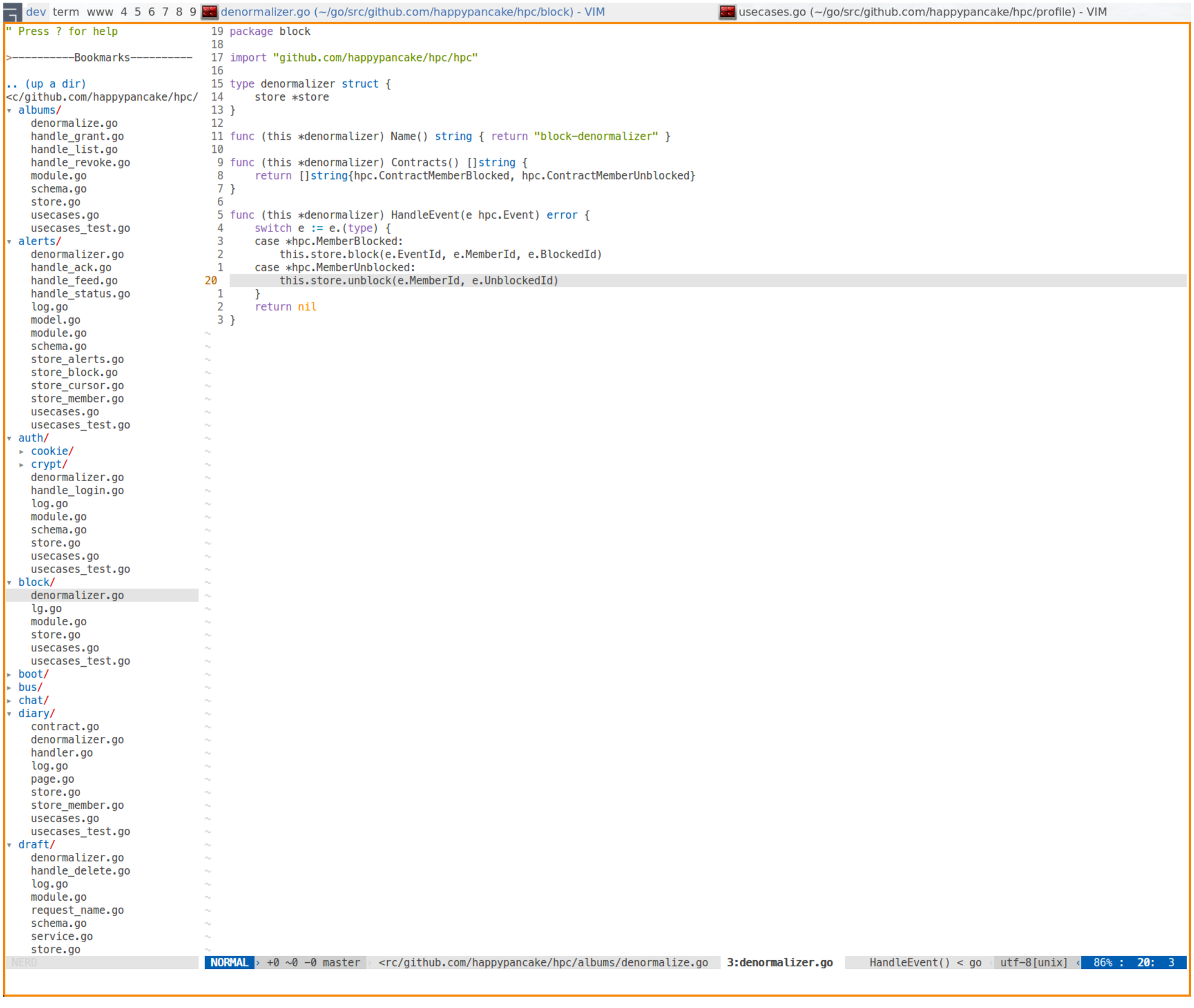


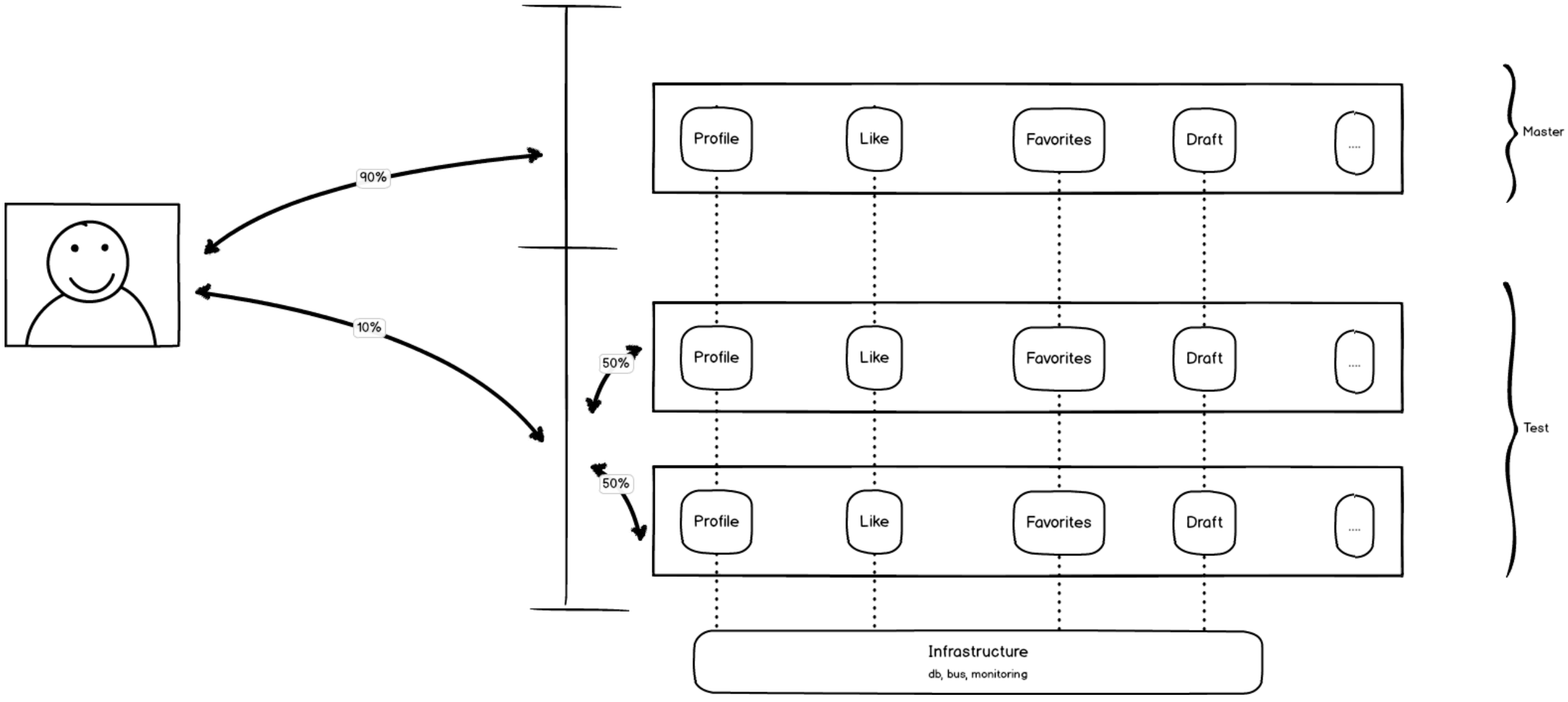
# Shipping

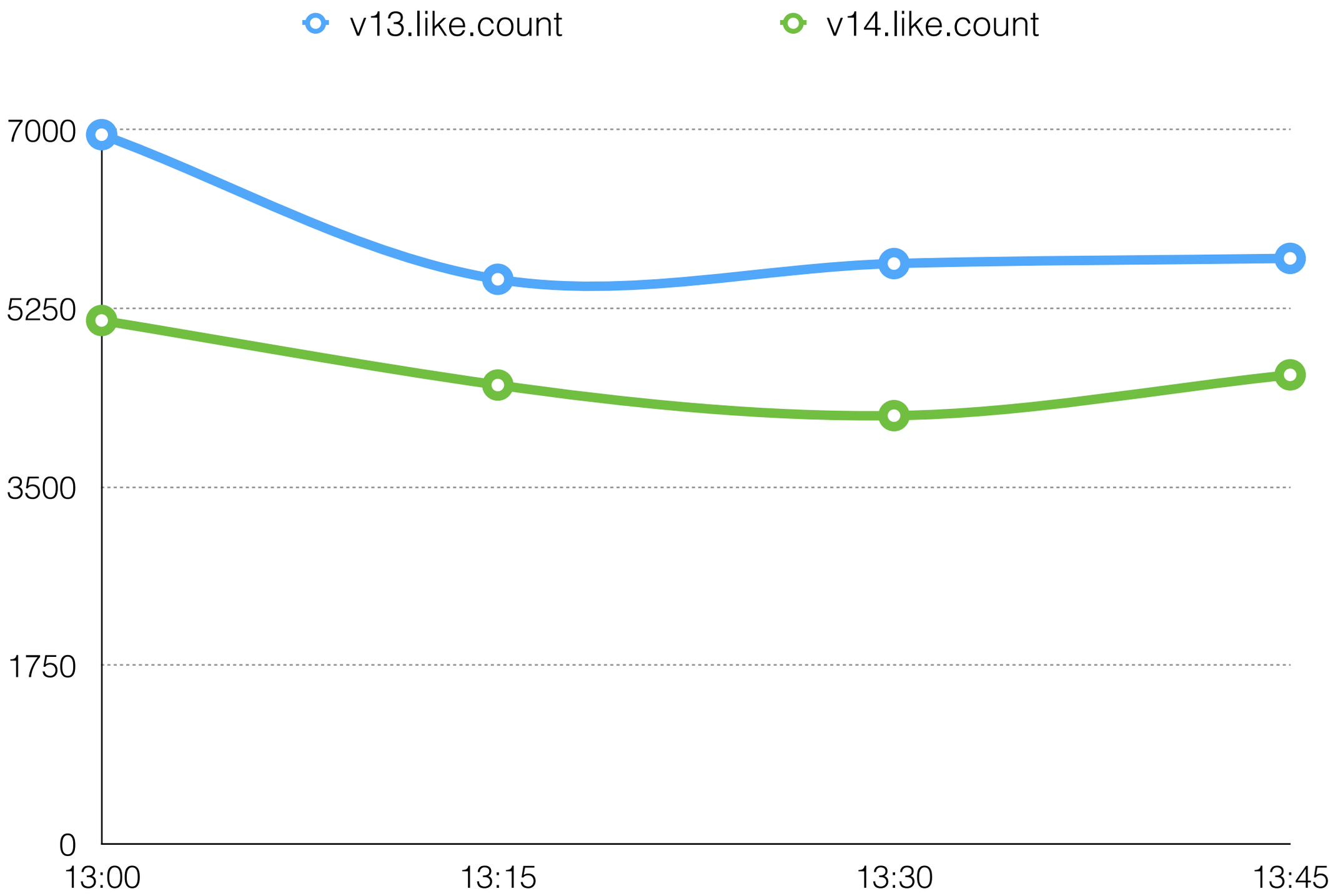

feature branches
develop
release branches
hotfixes
master
Time
Tag 0.1
Feature for future release
Major feature for next release
Severe bug fixed for production: hotfix 0.2
Incorporate bugfix in develop
Tag 0.2
Start of release branch for 1.0
From this point on, "next release" means the release after 1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop
Tag 1.0

“An µService doesn’t provide any value on its own”

“a μService is to μ to treat as a single product”

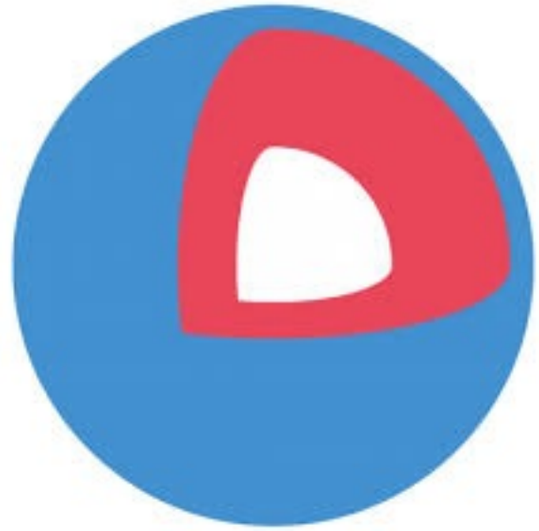

dev term www 4 5 6 7 8 9
denormalizer.go (~/go/src/github.com/happypancake/hpc/block) - VIM
usecases.go (~/go/src/github.com/happypancake/hpc/profile) - VIM
" Press ? for help
>----------Bookmarks----------
.. (up a dir)
<c/github.com/happypancake/hpc/
albums/
denormalize.go
handle_grant.go
handle_list.go
handle_revoke.go
module.go
schema.go
store.go
usecases.go
usecases_test.go
alerts/
denormalizer.go
handle_ack.go
handle_feed.go
handle_status.go
log.go
model.go
module.go
schema.go
store_alerts.go
store_block.go
store_cursor.go
store_member.go
usecases.go
usecases_test.go
auth/
cookie/
crypt/
denormalizer.go
handle_login.go
log.go
module.go
schema.go
store.go
usecases.go
usecases_test.go
block/
denormalizer.go
lg.go
module.go
store.go
usecases.go
usecases_test.go
boot/
bus/
chat/
diary/
contract.go
denormalizer.go
handler.go
log.go
page.go
store.go
store_member.go
usecases.go
usecases_test.go
draft/
denormalizer.go
handle_delete.go
log.go
module.go
request_name.go
schema.go
service.go
store.go
NERD
19 package block
18
17 import "github.com/happypancake/hpc/hpc"
16
15 type denormalizer struct {
14 store *store
13 }
12
11 func (this *denormalizer) Name() string { return "block-denormalizer" }
10
9 func (this *denormalizer) Contracts() []string {
8 return []string{hpc.ContractMemberBlocked, hpc.ContractMemberUnblocked}
7 }
6
5 func (this *denormalizer) HandleEvent(e hpc.Event) error {
4 switch e := e.(type) {
3 case *hpc.MemberBlocked:
2 this.store.block(e.EventId, e.MemberId, e.BlockedId)
1 case *hpc.MemberUnblocked:
20 this.store.unblock(e.MemberId, e.UnblockedId)
1 }
2 return nil
3 }
NORMAL +0 ~0 -0 master <rc/github.com/happypancake/hpc/albums/denormalize.go 3:denormalizer.go HandleEvent() < go utf-8[unix] 86% : 20: 3


90%
10%
50%
50%
Profile
Like
Favorites
Draft
....
Profile
Like
Favorites
Draft
....
Profile
Like
Favorites
Draft
...
Infrastructure
db, bus, monitoring
Master
Test


v13.like.count
v14.like.count
7000
5250
3500
1750
0
13:00
13:15
13:30
13:45

# ETCD

- Simple, curl-able API (HTTP + JSON)
- Benchmarked 1000s of writes/s per instance
- Properly distributed using Raft protocol
- Atomic test and set
- Easily listen for changes to a prefix via HTTP long-polling



# Summary

# You've been a great audience!

*already knew that when creating this slide*