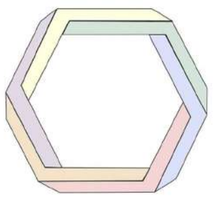
The Microservice Hypothesis

Cesare Pautasso
http://www.pautasso.info/
c.pautasso@ieee.org

Keynote
International Conference on Microservices 2022
May 12, 2022 (Paris, France)

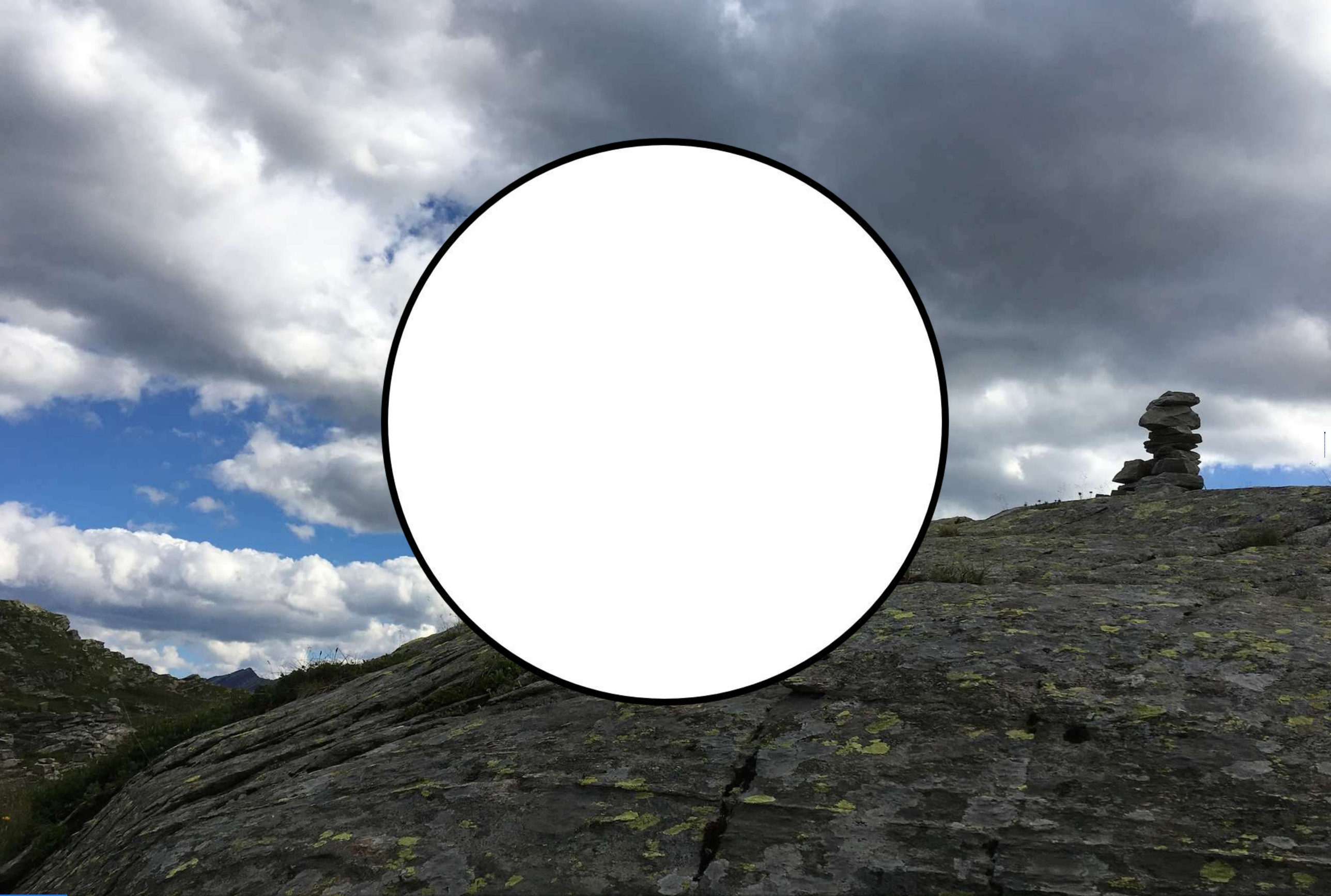


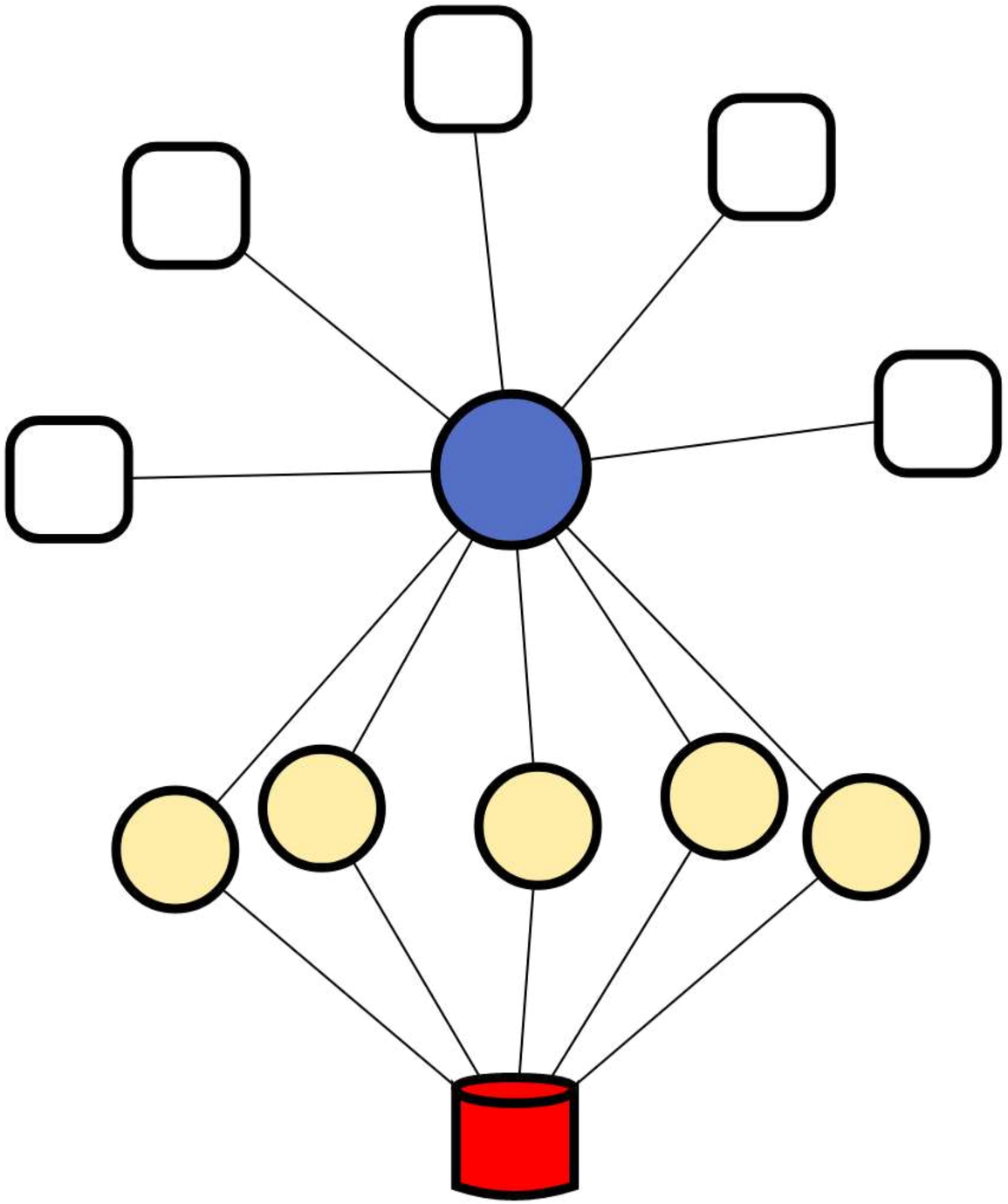
Contents

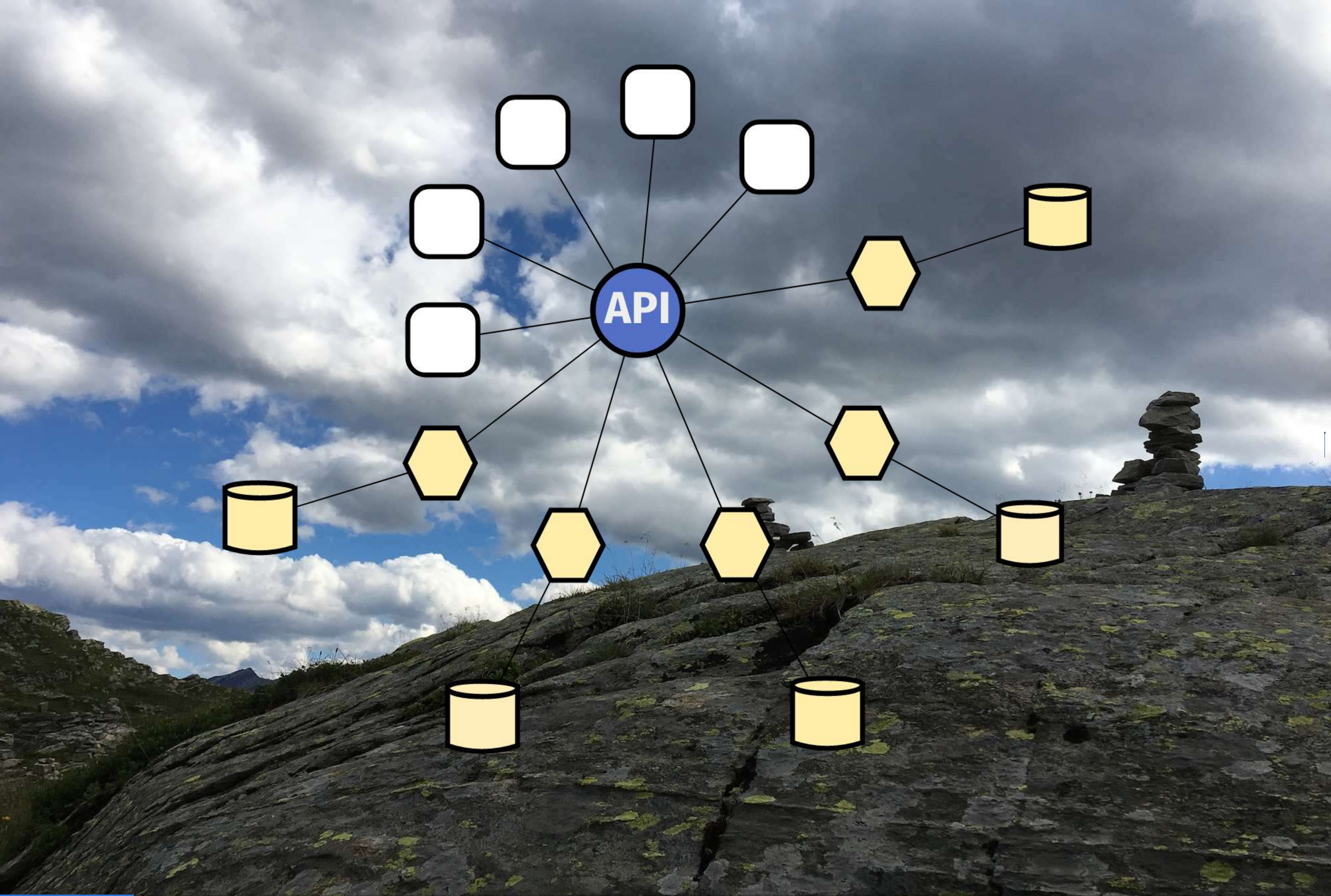
- Splitting the Monolith
- API Visualization
- API Design and Evolution Patterns
- The Microservice Hypothesis

Abstract

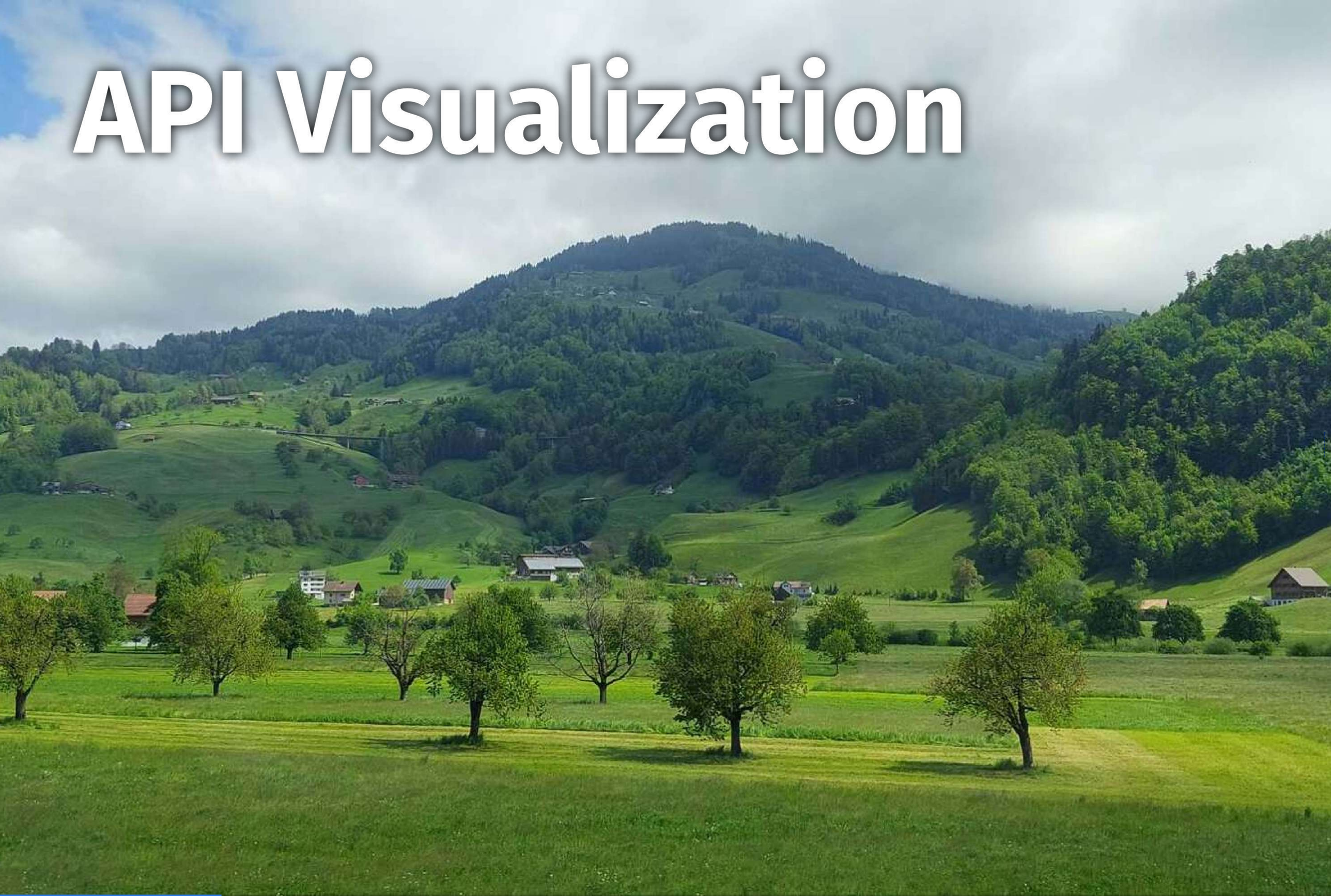
Microservice architectures are based on the hypothesis that by keeping components small, this can foster their autonomy and independent evolution. Microservices reframed the modularity issue as a decomposition problem, where APIs are introduced to reconnect separately deployed components. In this talk we will explore a large collection of real-world APIs looking for patterns attempting to illustrate the relationship between size and coupling, both in space and time.






API

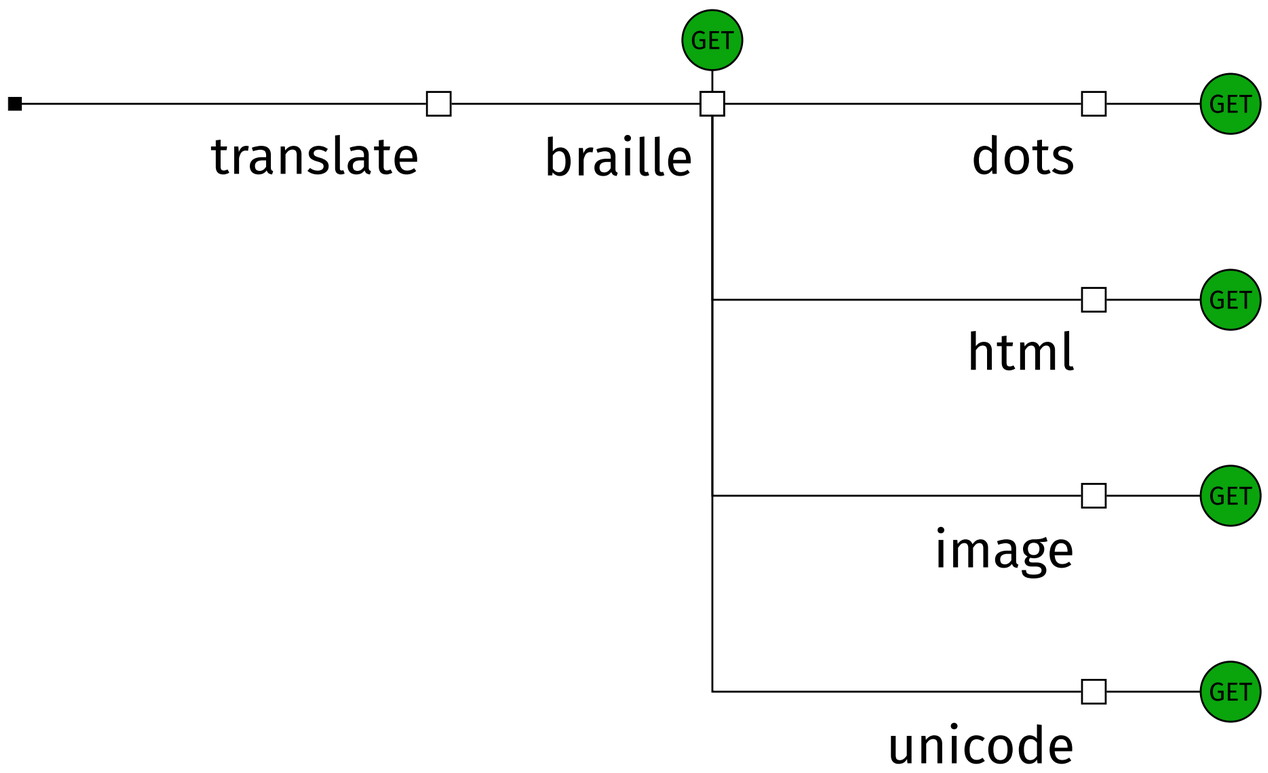
API Visualization



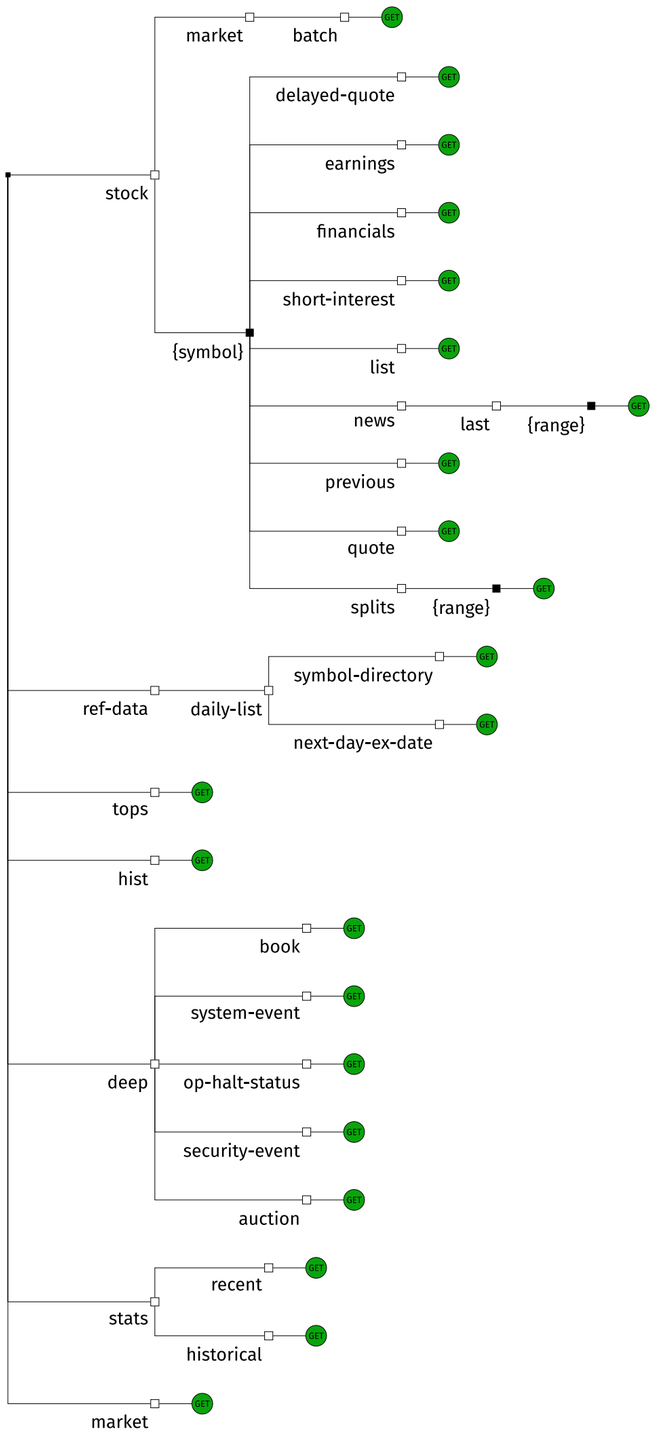
API Design and Evolution Patterns

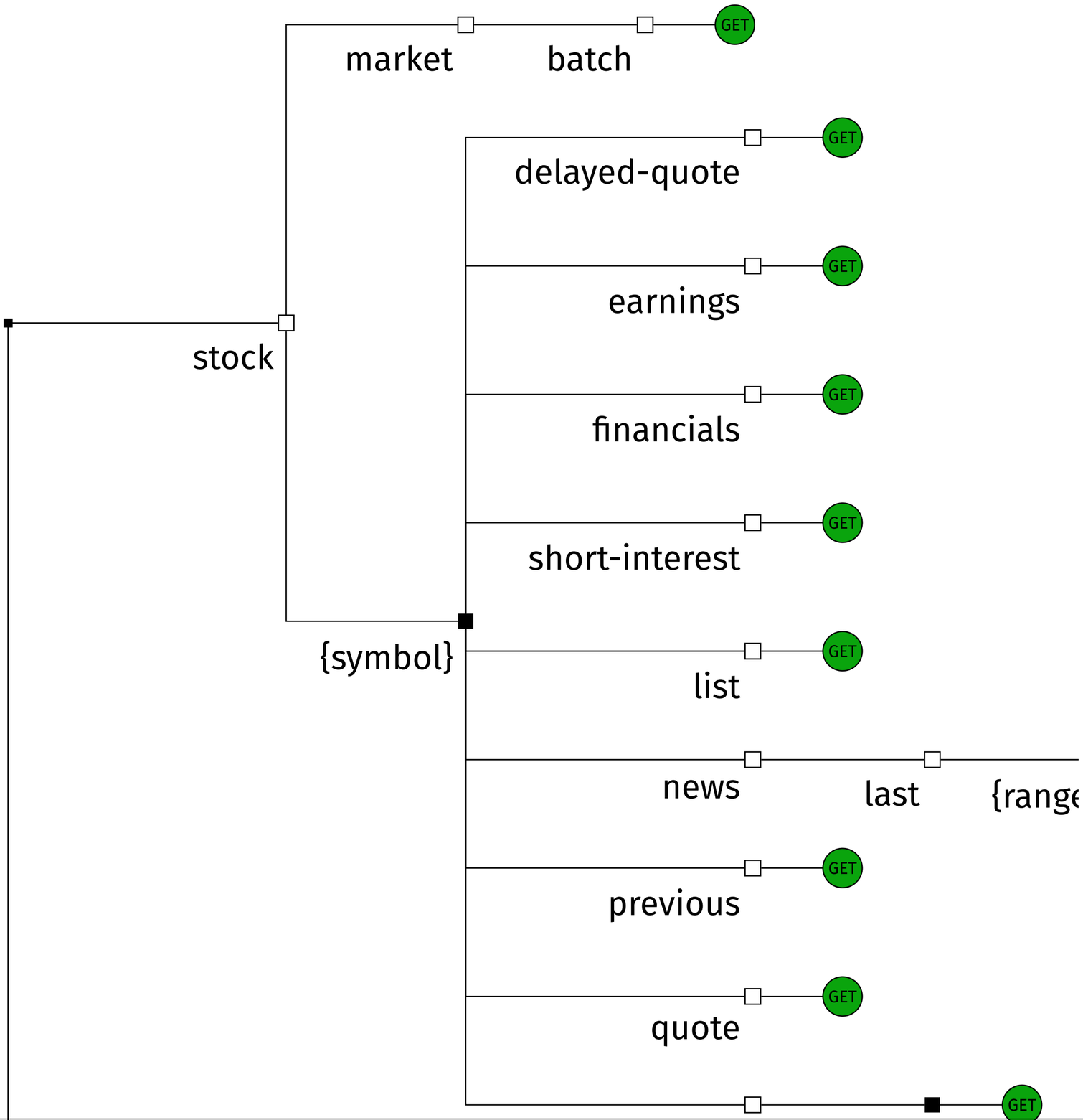
Operations = Resource Path + Method

Multiple Representations

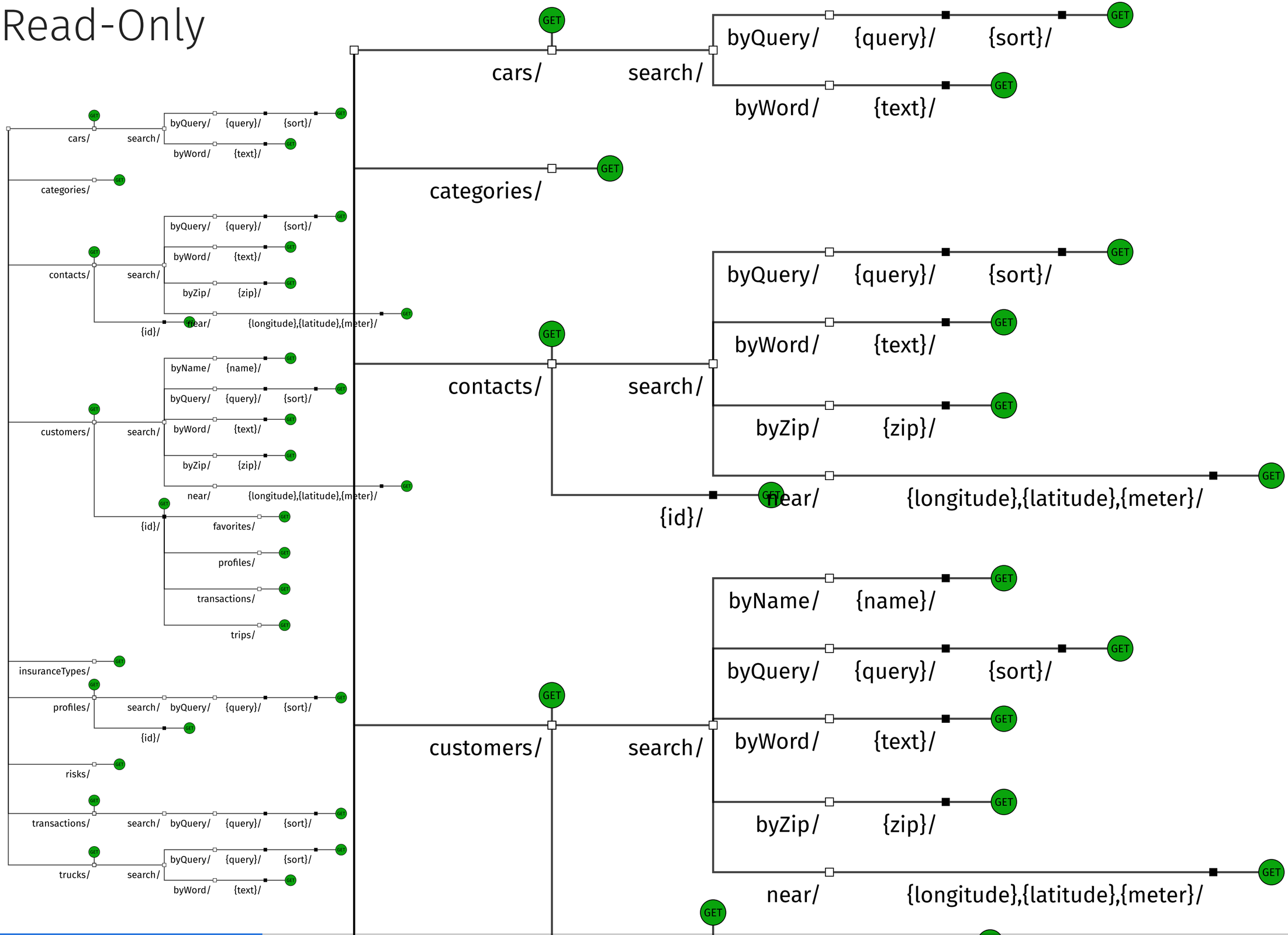


Read-Only


market
batch
GET
delayed-quote
earnings
financials
short-interest
stock
{symbol}
list
news
last
{range}
previous
quote
splits
{range}
symbol-directory
ref-data
daily-list
next-day-ex-date
tops
hist
book
system-event
deep
op-halt-status
security-event
auction
recent
stats
historical
market


market
batch
GET
delayed-quote
earnings
financials
short-interest
stock
{symbol}
list
news
last
{range
previous
quote

Read-Only

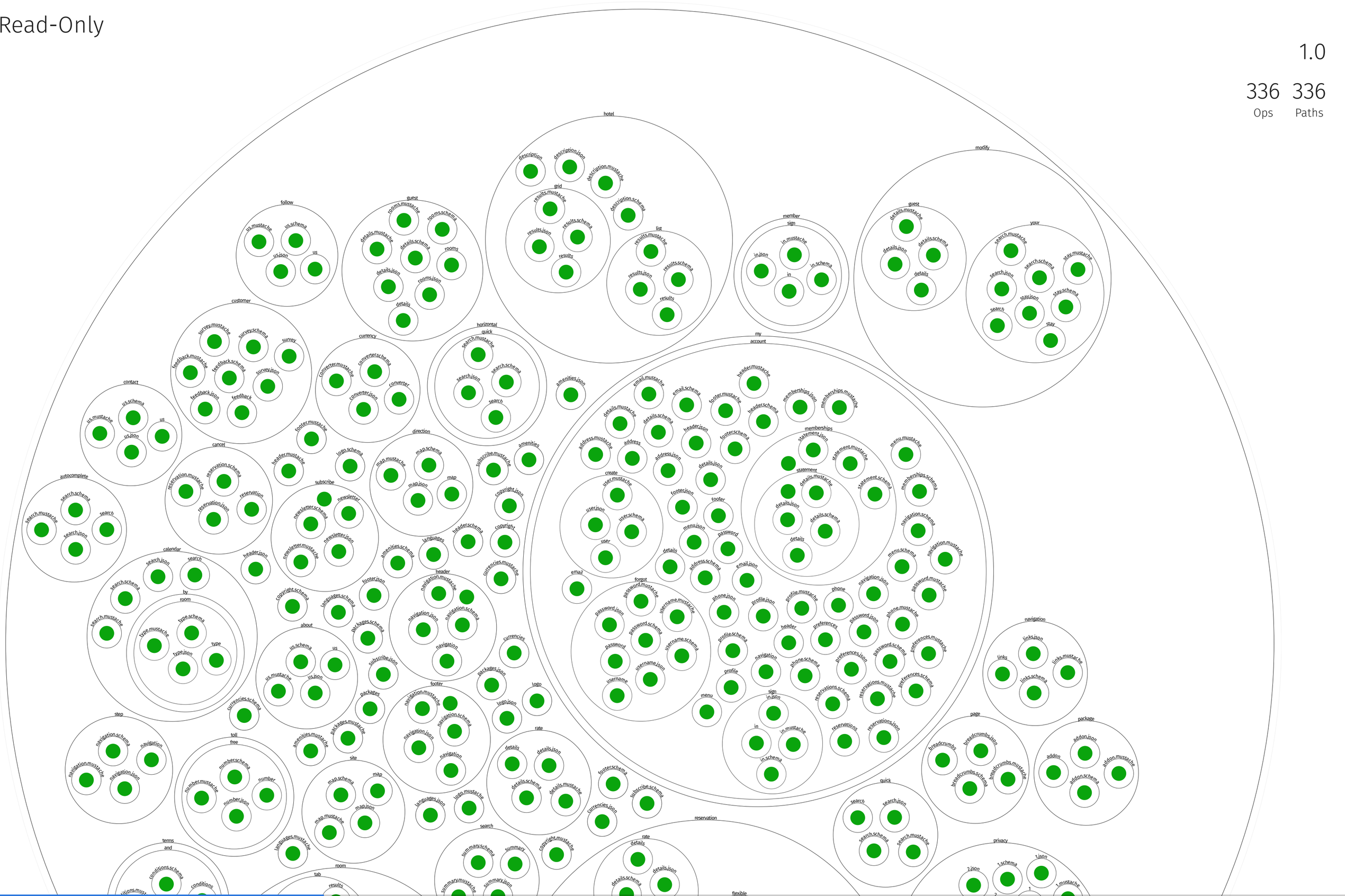

cars/
search/
byQuery/
{query}/
{sort}/
byWord/
{text}/
categories/
contacts/
search/
byQuery/
{query}/
{sort}/
byWord/
{text}/
byZip/
{zip}/
{id}/
near/
{longitude},{latitude},{meter}/
customers/
search/
byName/
{name}/
byQuery/
{query}/
{sort}/
byWord/
{text}/
byZip/
{zip}/
near/
{longitude},{latitude},{meter}/
{id}/
favorites/
profiles/
transactions/
trips/
insuranceTypes/
profiles/
search/
byQuery/
{query}/
{sort}/
{id}/
risks/
transactions/
search/
byQuery/
{query}/
{sort}/
trucks/
search/
byQuery/
{query}/
{sort}/
byWord/
{text}/
GET

Read-Only

1.0

336 Ops

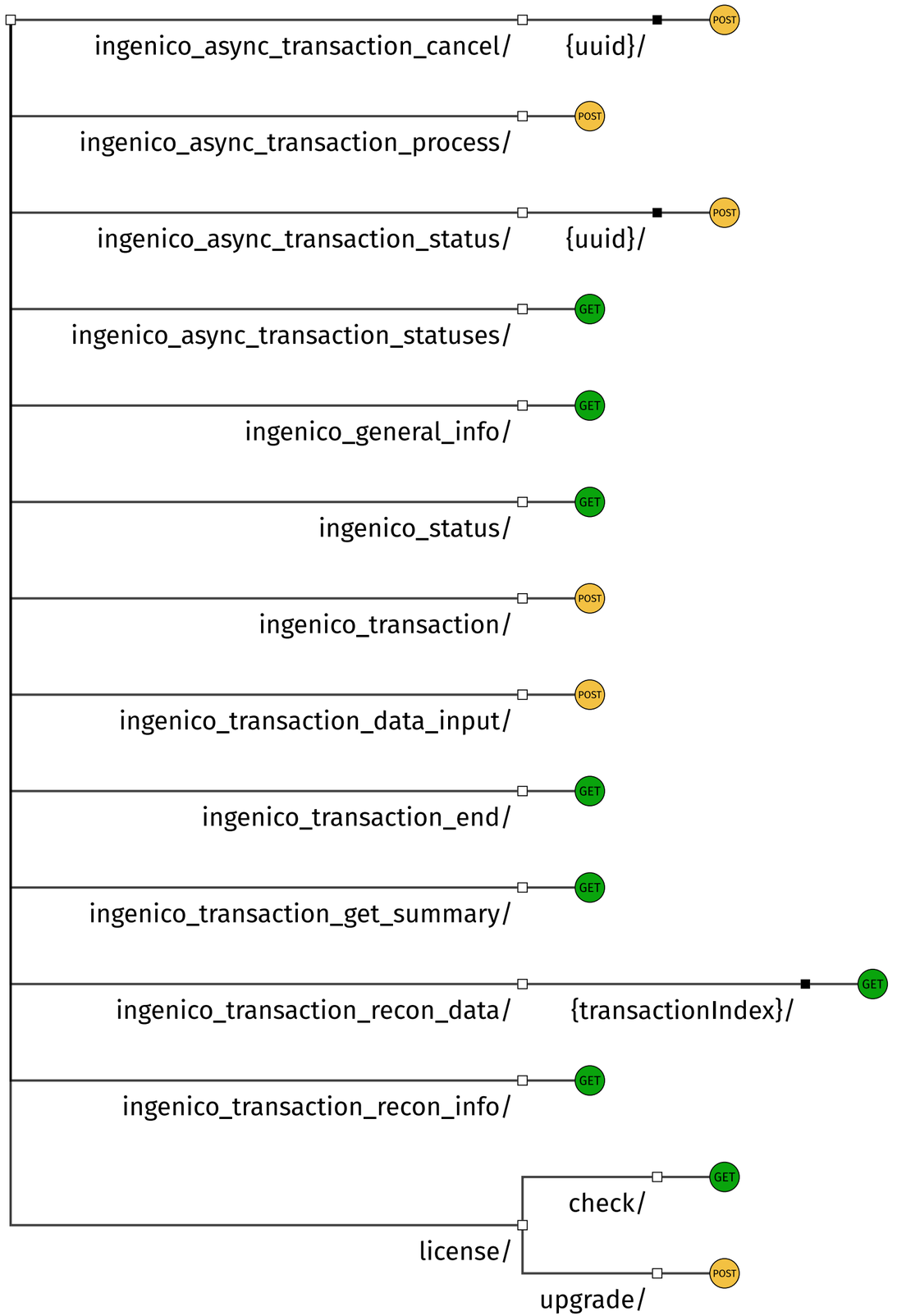
336 Paths



Read-Write


GET
GetPerVMTypeOneBootShutDownData
GET
getAllVMTypesBootShutDownDataAvg
GET
getAllVMTypesBootShutDownDataRegression
GET
getPerVMTypeAllBootShutDownData
GET
getandStoreRegressionValues
POST
initUserConfig
GET
listAllInstances
GET
trainDataSetRegression

Read-Write

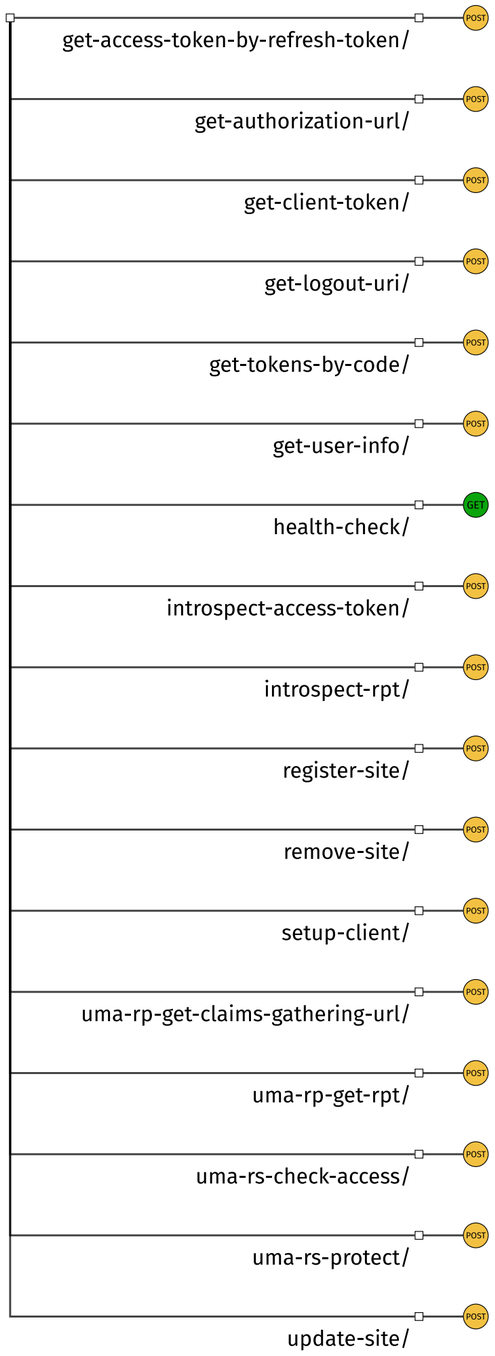

POST
ingenico_async_transaction_cancel/
{uuid}/
POST
ingenico_async_transaction_process/
POST
ingenico_async_transaction_status/
{uuid}/
GET
ingenico_async_transaction_statuses/
GET
ingenico_general_info/
GET
ingenico_status/
POST
ingenico_transaction/
POST
ingenico_transaction_data_input/
GET
ingenico_transaction_end/
GET
ingenico_transaction_get_summary/
GET
ingenico_transaction_recon_data/
{transactionIndex}/
GET
ingenico_transaction_recon_info/
GET
check/
license/
POST
upgrade/

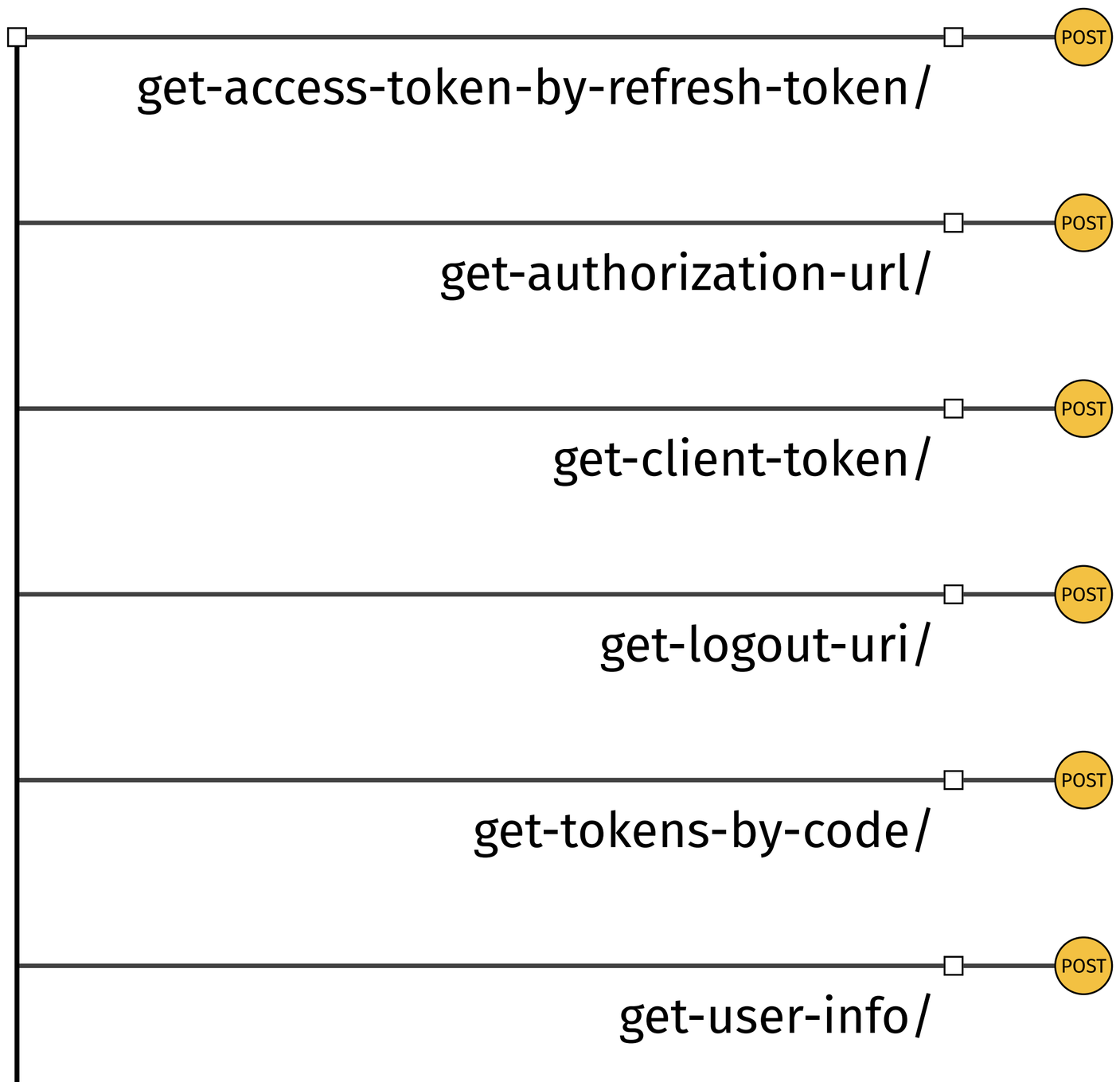
Read-Write

GOXD
beta

335 Ops
335 Paths

Remote Procedure Call

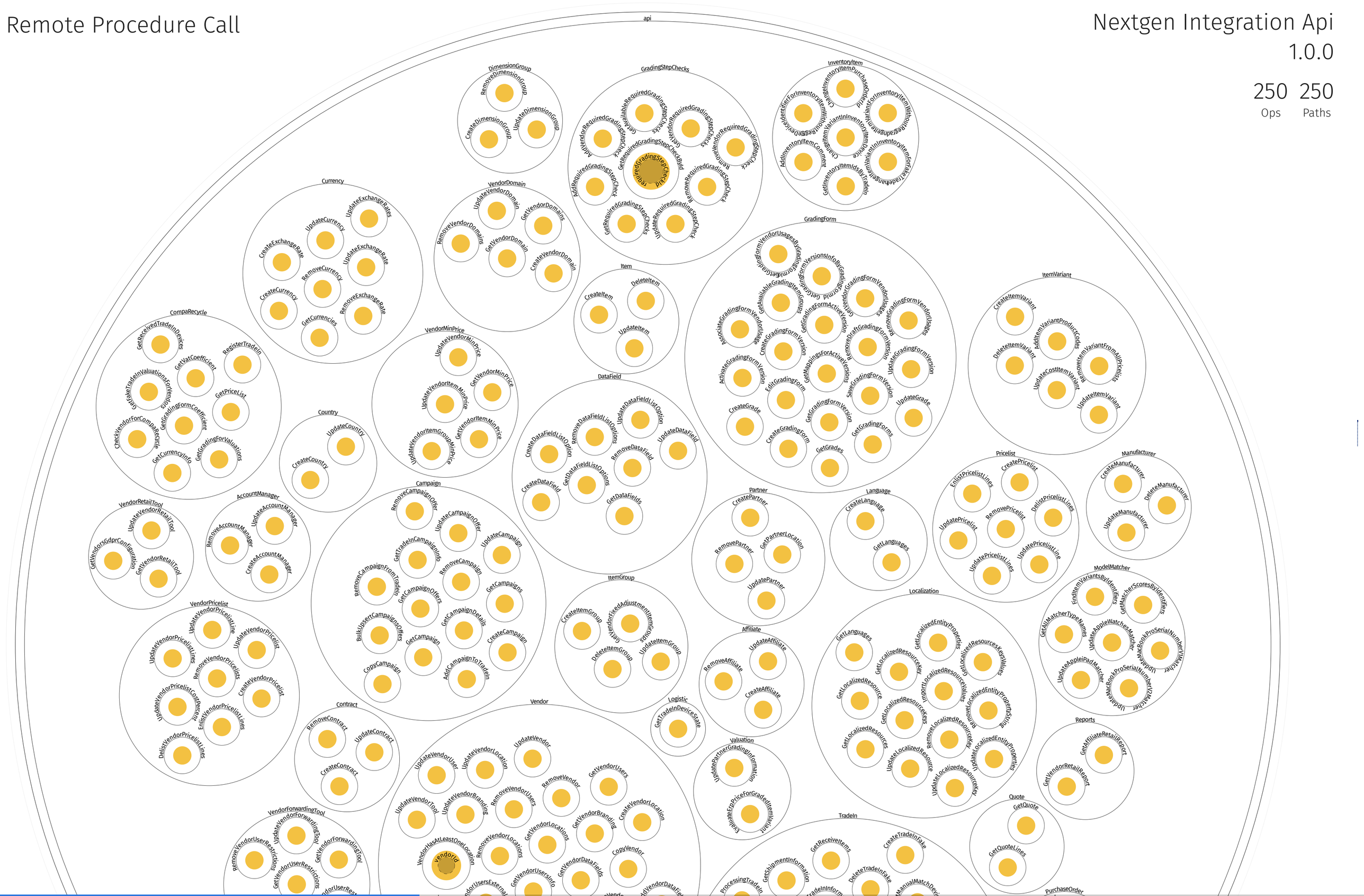




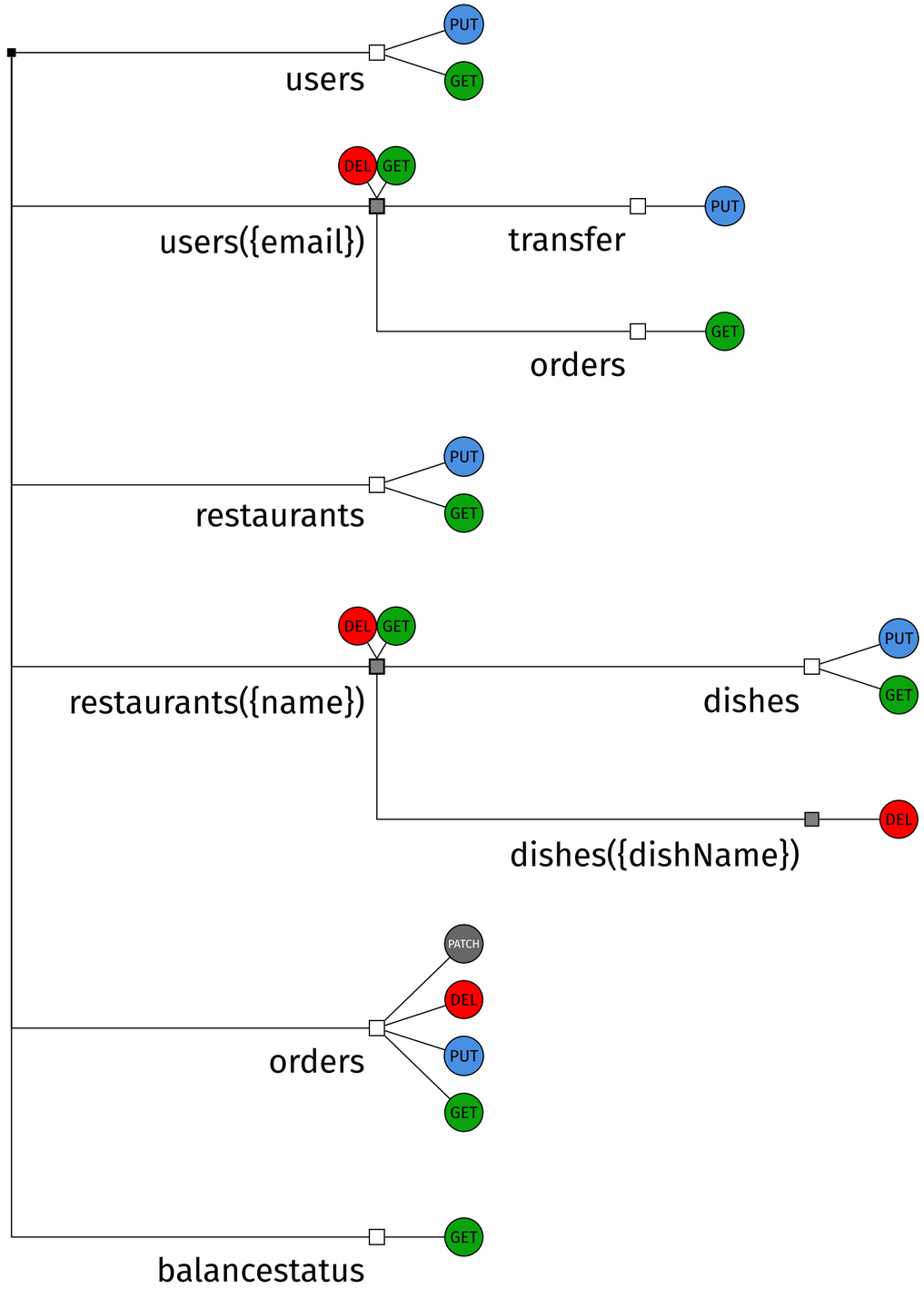
Remote Procedure Call

Nextgen Integration Api
1.0.0

250 Ops 250 Paths



POST-less


PUT
GET
users
DEL
GET
PUT
users({email})
transfer
GET
orders
PUT
GET
restaurants
DEL
GET
PUT
restaurants({name})
dishes
DEL
dishes({dishName})
PATCH
DEL
PUT
GET
orders
GET
balancestatus

METHOD .../action

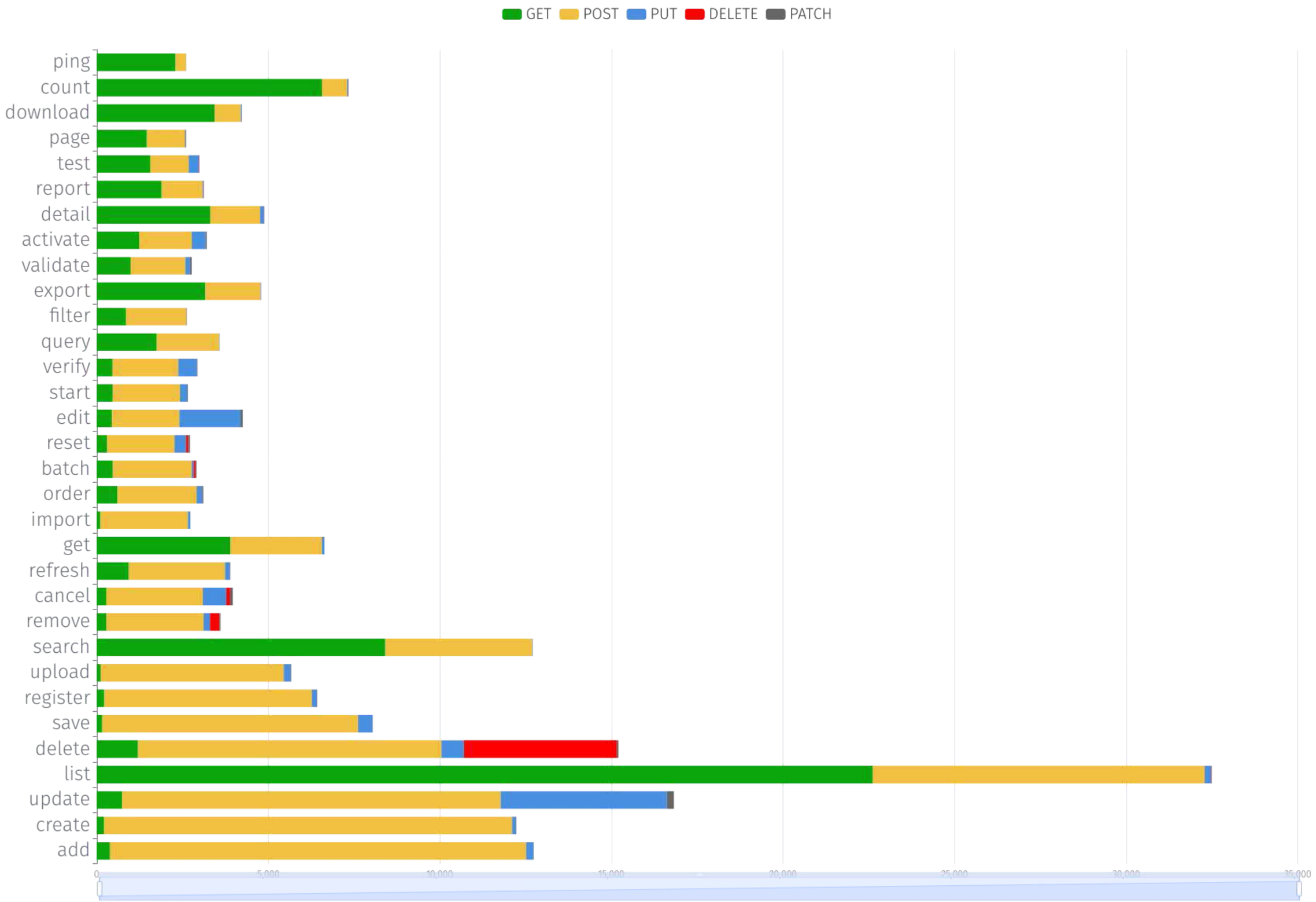
Processing Resource

How can an API provider allow its remote clients to trigger actions in it?

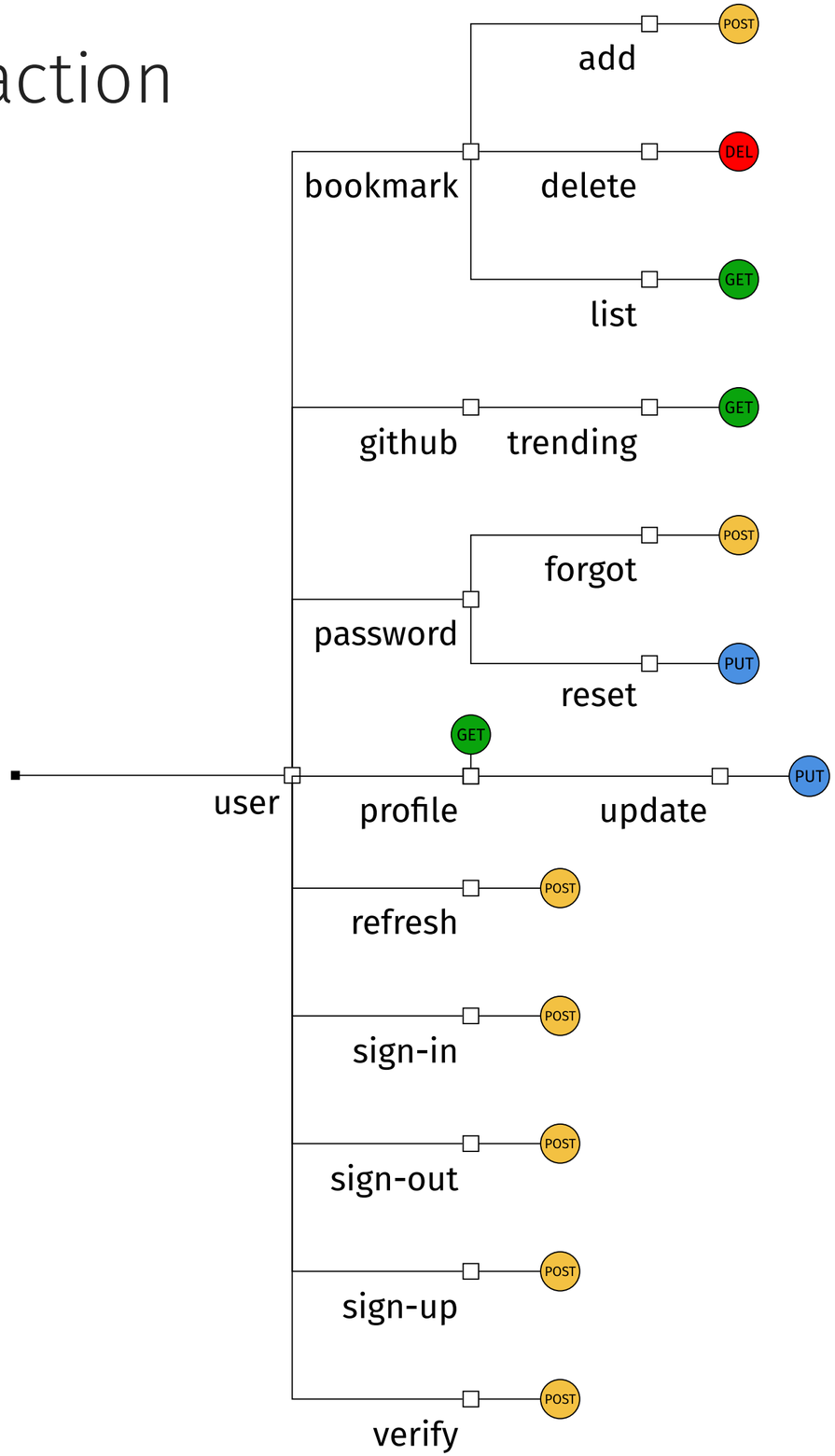
Actions → Methods

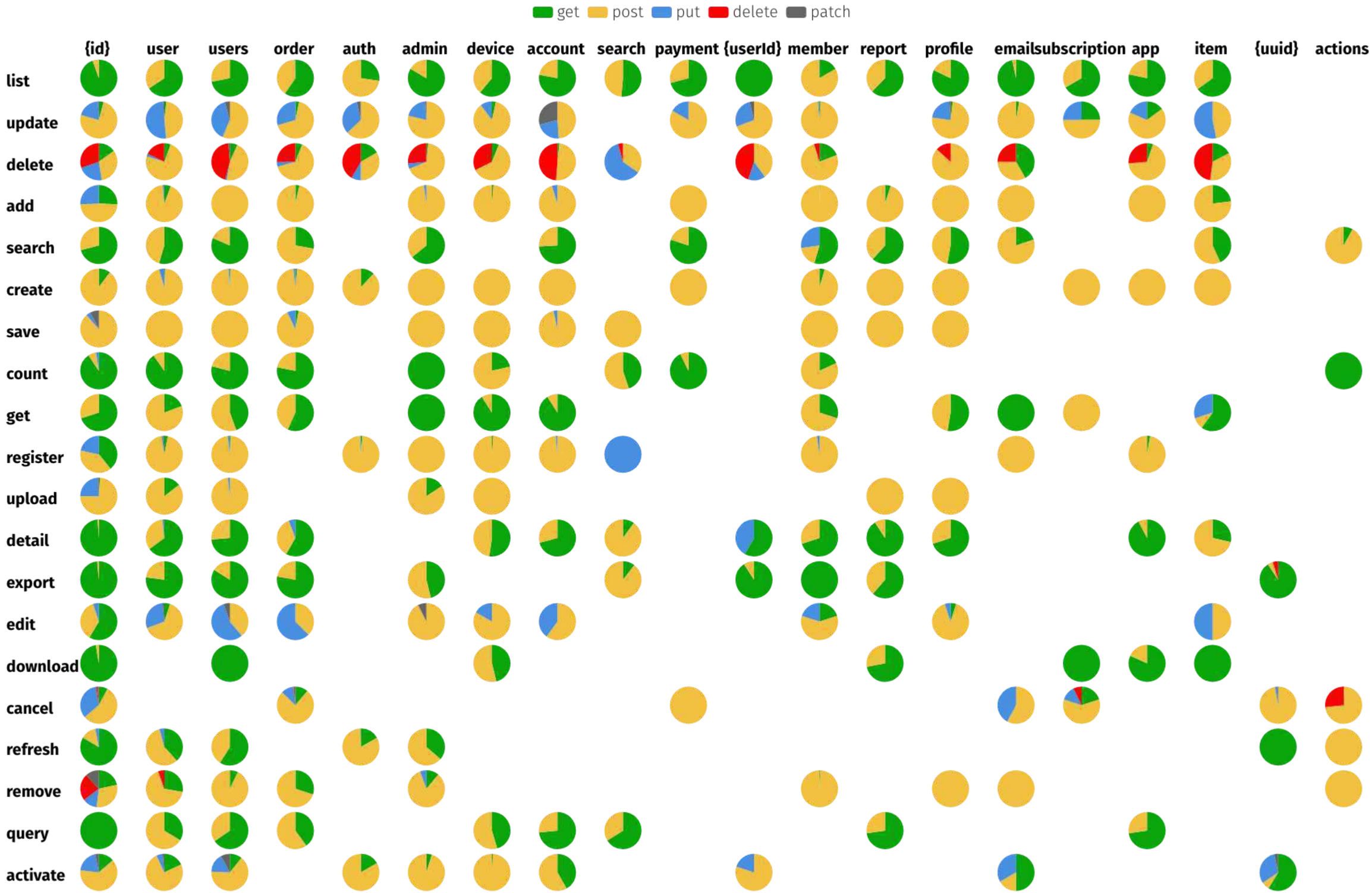
list, update, delete, add, search, create, save, count, get, register, upload, detail, export, edit, download, cancel, refresh, remove, query, activate, report, order, test, verify, batch, validate, import, reset, start, filter, ping, page, send, check, view, profile, confirm, state, bulk, disable, email, enable, preview, account, submit, sync, scheme, finish, insert, request, archive, authenticate, index, invite, inventory, score, deactivate, image, notify, stop, publish, log, comment, read, revoke, file, find, approve, change, last, content, audit, complete, snooze, accept, set, contour, range, generate, message, copy, close, grant, apply, run, refund, address, modify, reject, restore, clone, project, default, excel, contact, install, balance, name, restart, subscribe, result, group, trace, clear, pay, move, map, code, company, process, push, share, tree, ticket, traffic, schedule, resume, post, type, sign, pause, control, assign, issue, finalize, unlock, discover, catalog, lock, join, paginate, review, like, fetch, authorize, replace, select, execute, load, decline, phone, home, rename, dump, tag, transfer, calculate, resolve, stream, top, open, total, document, merge, invoice, form, alert, link, purchase, bind, connect, feed, show, google, record, trigger, reorder, key, reboot, card, charge, withdraw, store, duplicate, cache, block, source, action, swagger, release, time, terminate, task, campaign, commit, redeem, team, put, service, receive,

GET,
POST,
PUT,
DELETE,
PATCH,
OPTIONS


GET
POST
PUT
DELETE
PATCH
ping
count
download
page
test
report
detail
activate
validate
export
filter
query
verify
start
edit
reset
batch
order
import
get
refresh
cancel
remove
search
upload
register
save
delete
list
update
create
add
0
5,000
10,000
15,000
20,000
25,000
30,000
35,000

METHOD .../object/action

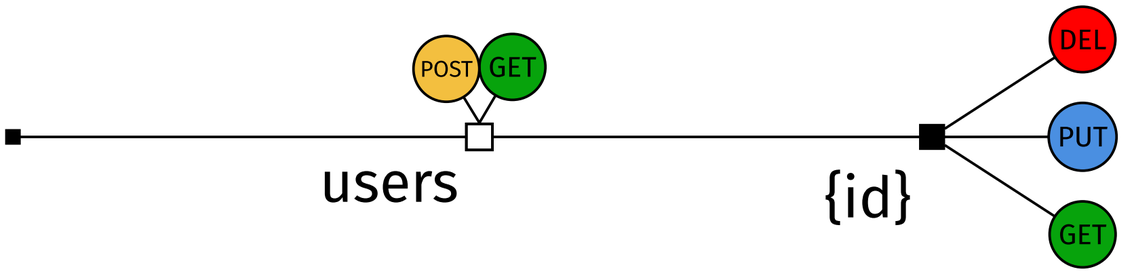


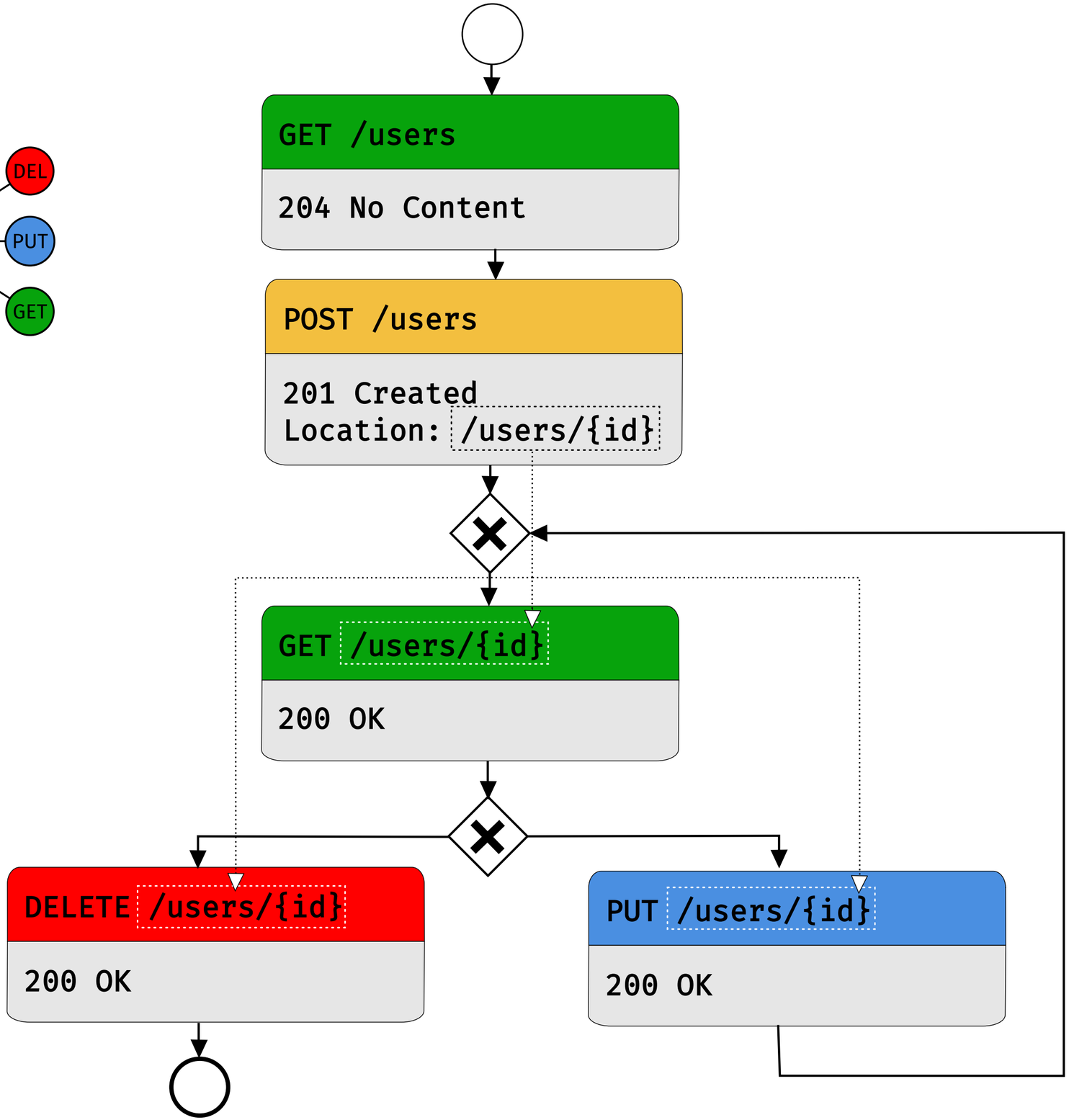

get
post
put
delete
patch
{id}
user
users
order
auth
admin
device
account
search
payment
{userId}
member
report
profile
emailsubscription
app
item
{uuid}
actions
list
update
delete
add
search
create
save
count
get
register
upload
detail
export
edit
download
cancel
refresh
remove
query
activate

Information Holder

How can an API expose data entities so that API clients can access and/or modify these entities concurrently without compromising data integrity and quality?

Collection


POST
GET
users
{id}
DEL
PUT
GET


GET /users
204 No Content
POST /users
201 Created
Location: /users/{id}
GET /users/{id}
200 OK
DELETE /users/{id}
200 OK
PUT /users/{id}
200 OK

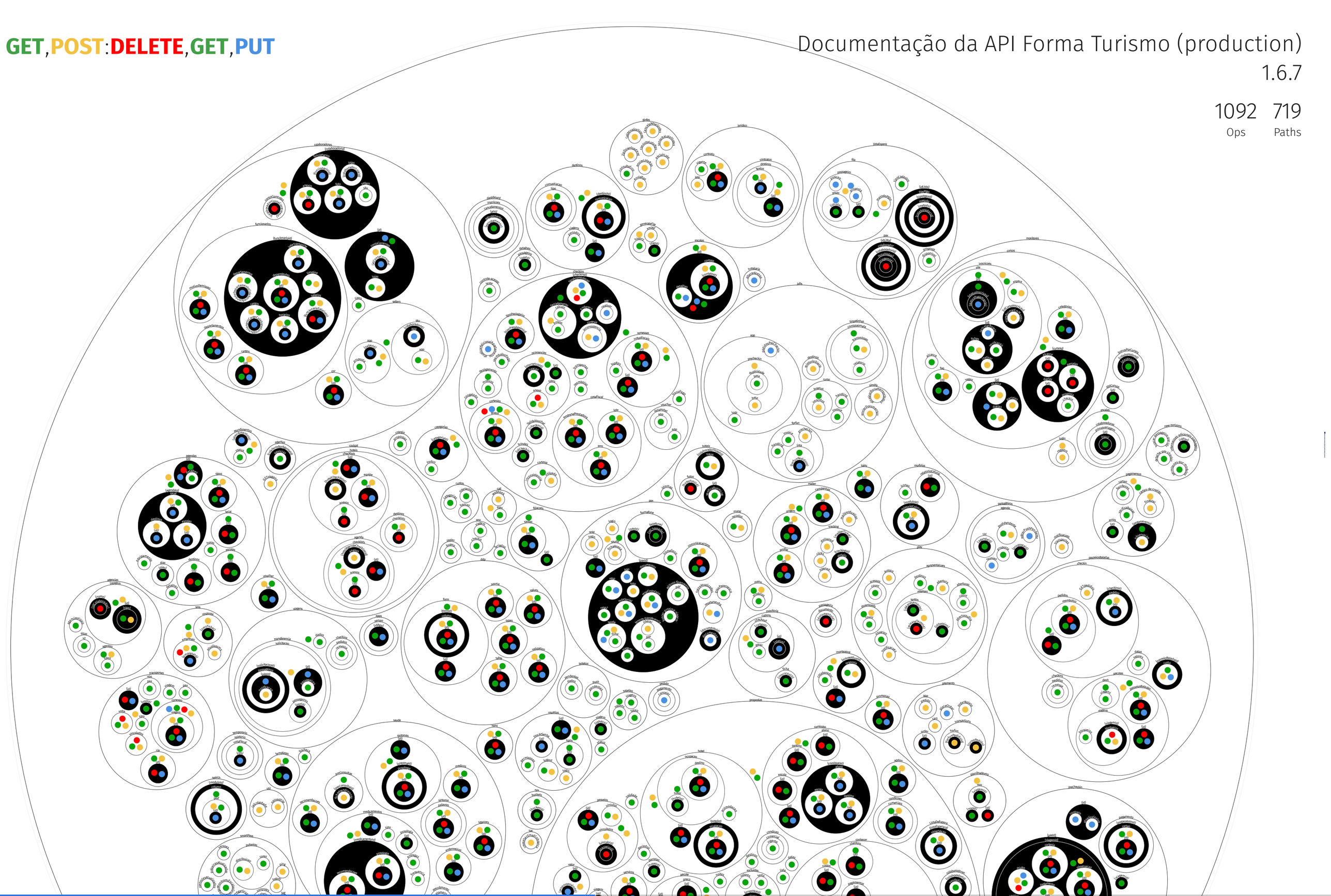
GET,POST:DELETE,GET,PUT

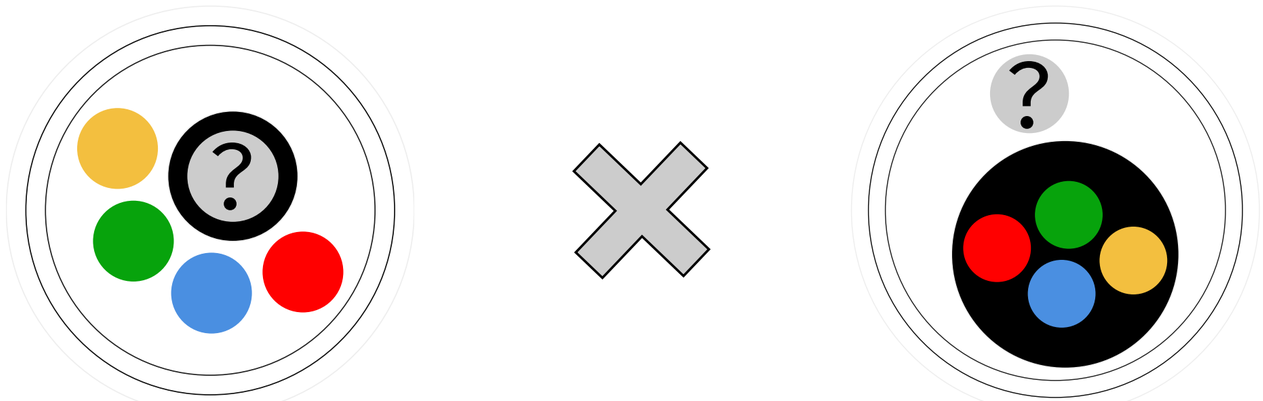
Documentação da API Forma Turismo (production)

1.6.7

1092 Ops

719 Paths

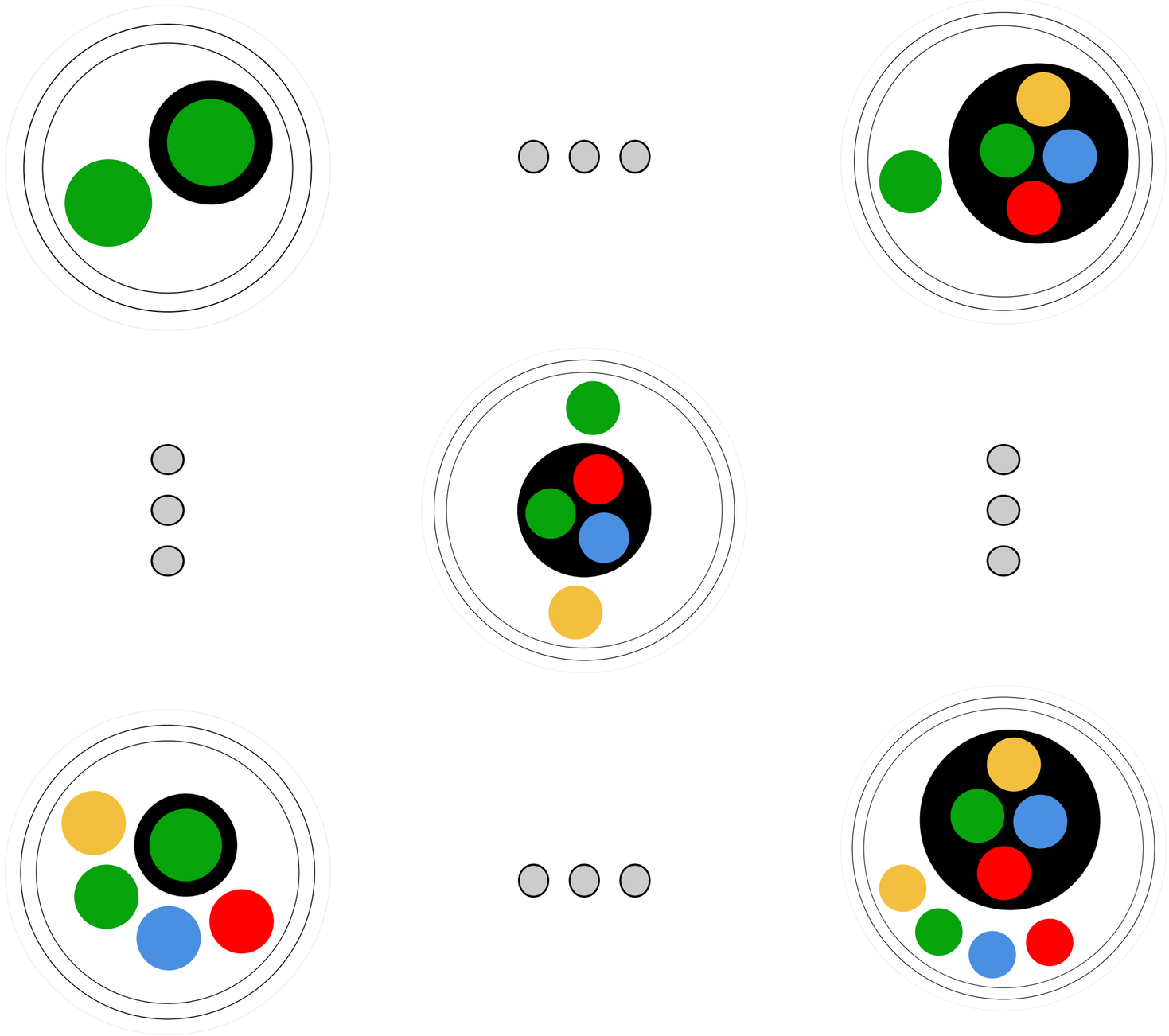


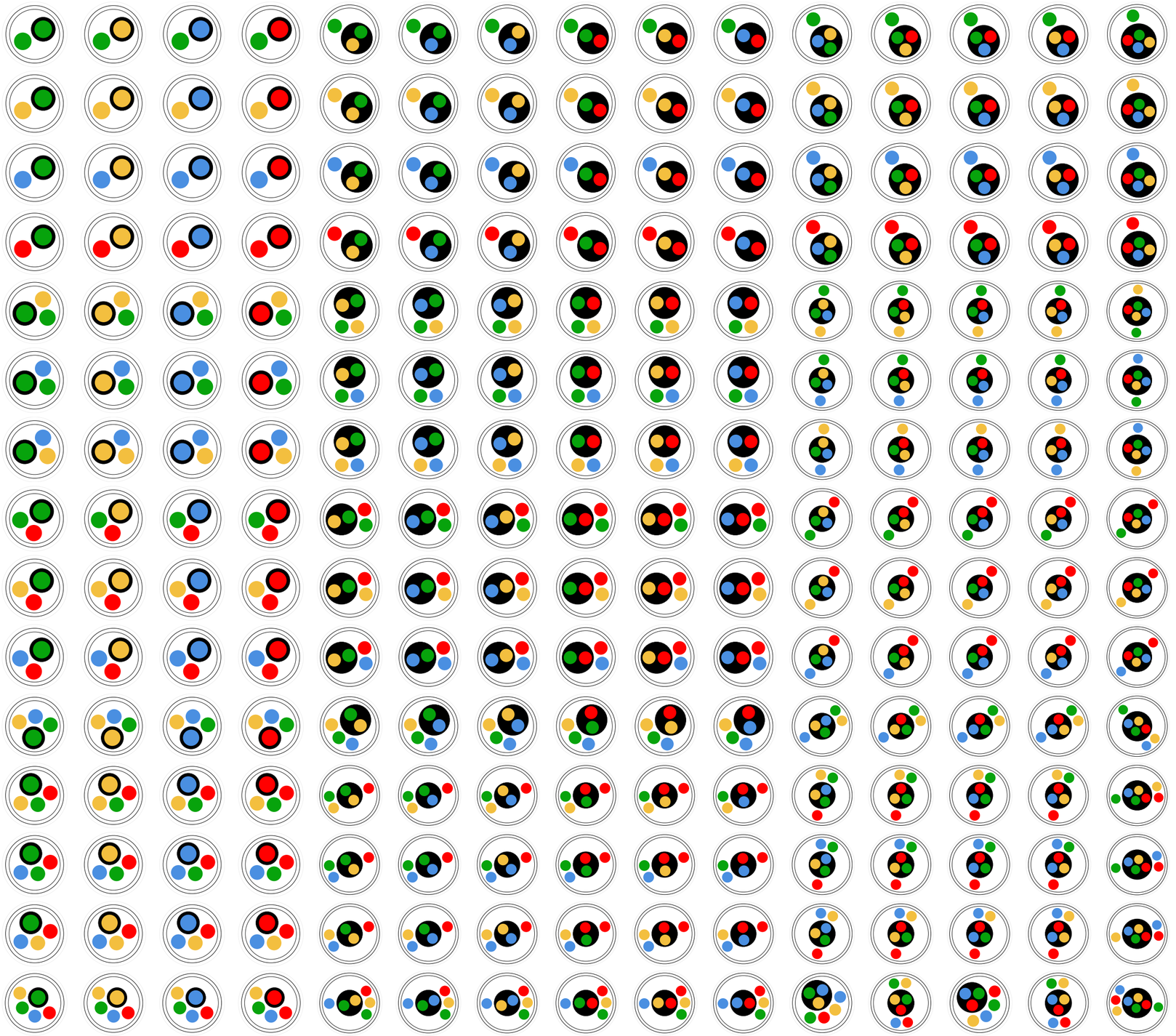


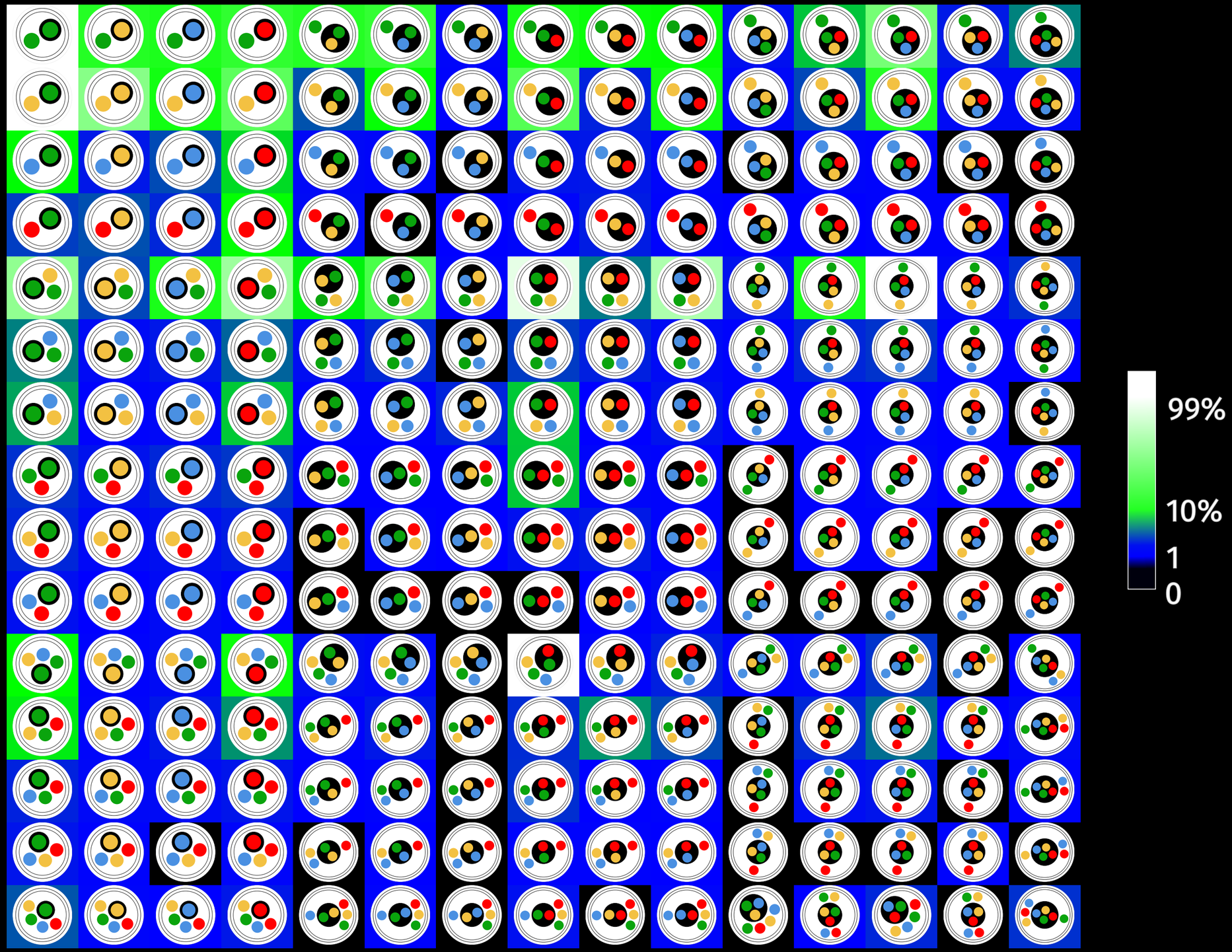
Container methods

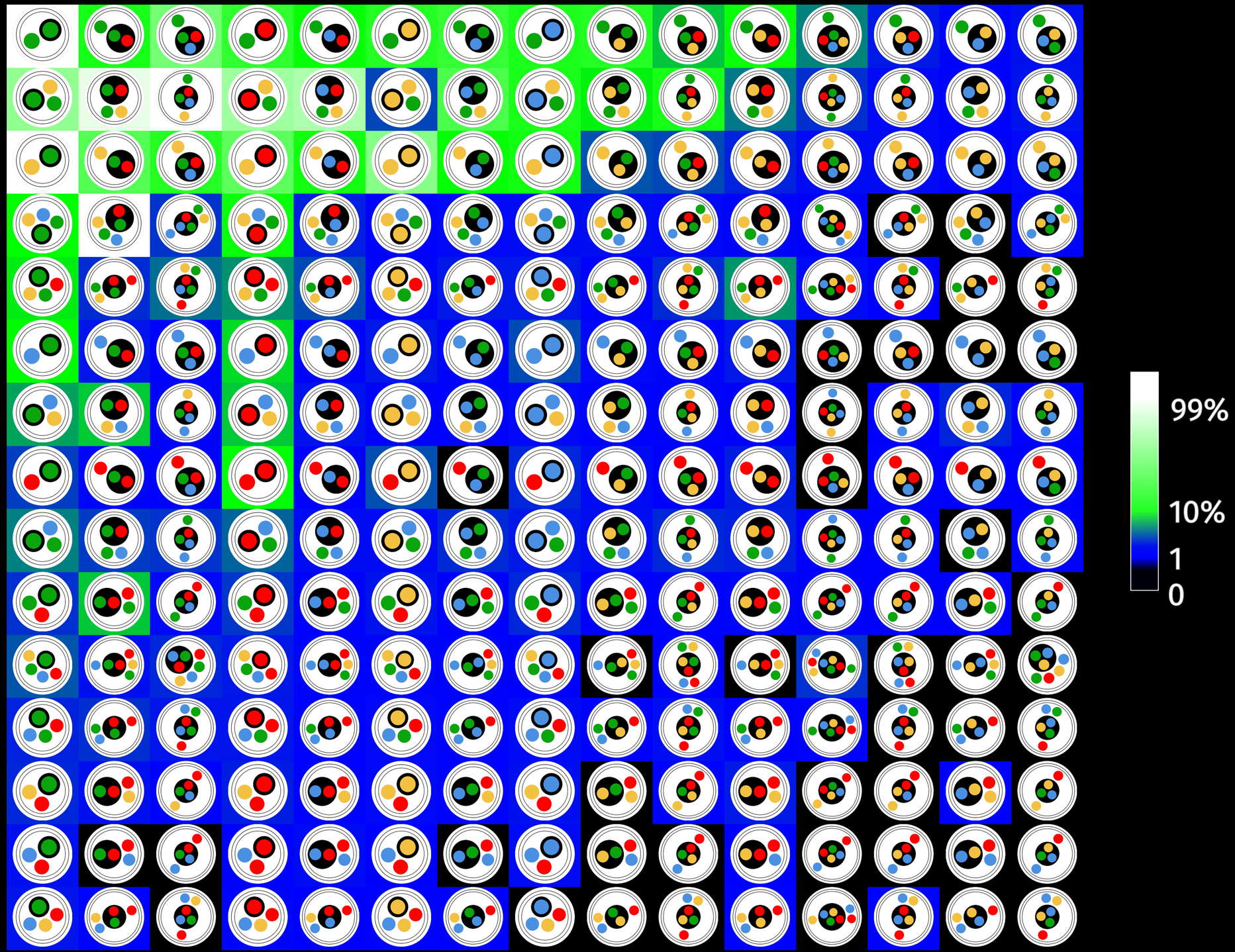
×

Item methods






99%
10%
1
0

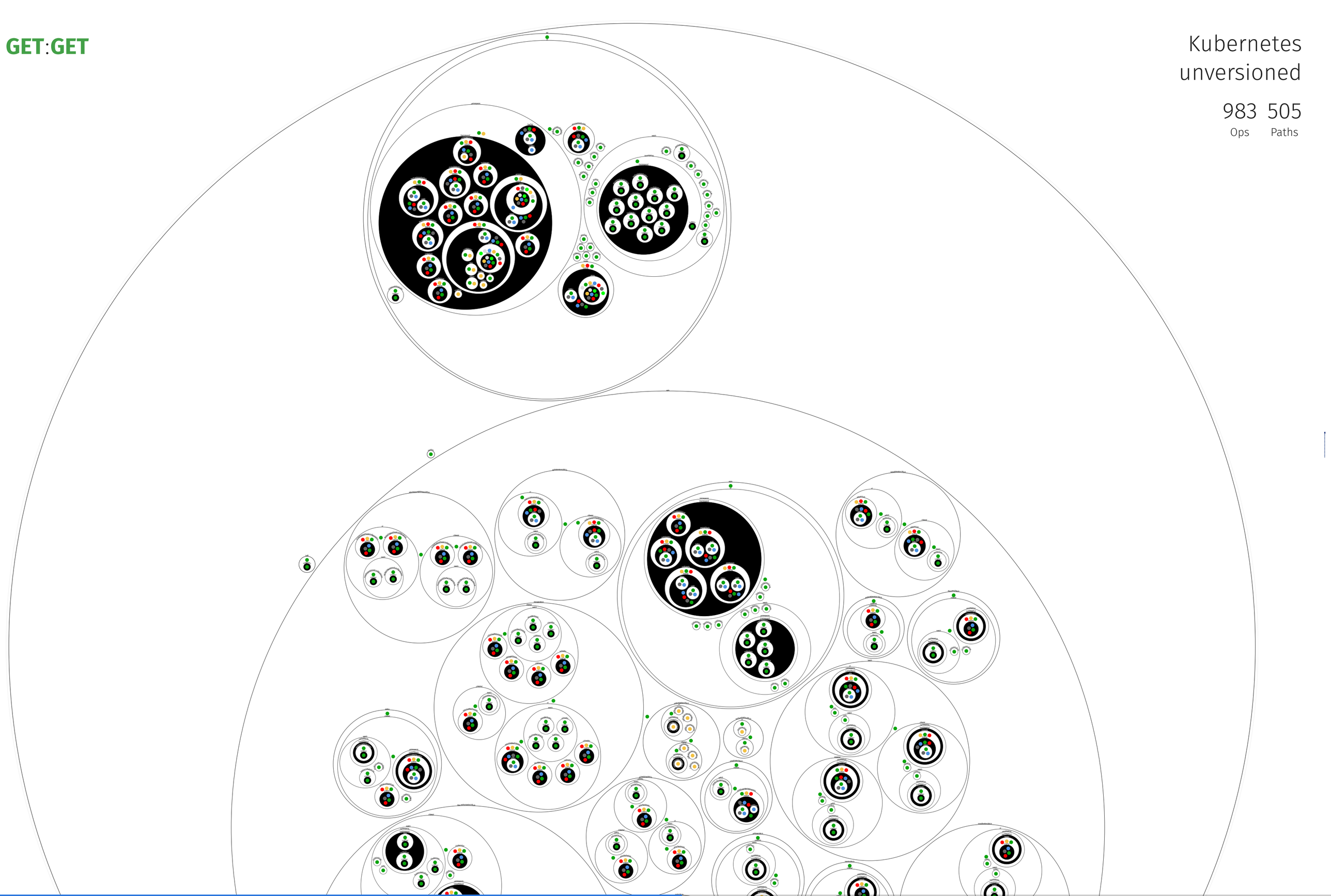

99%
10%
1
0

GET:**GET**

Kubernetes
unversioned

983 Ops

505 Paths



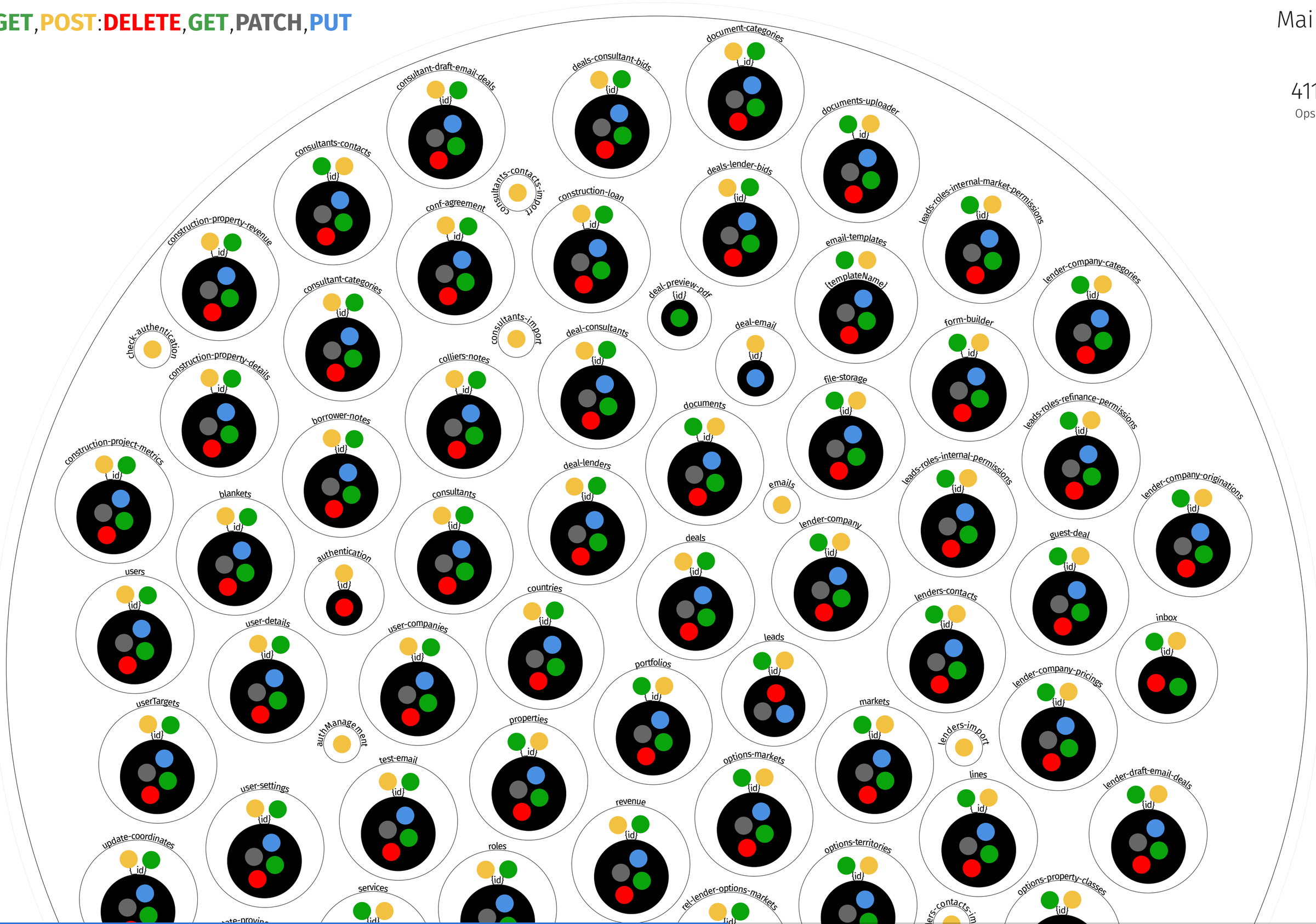
GET,POST:DELETE,GET,PATCH,PUT

Main API

0.0.1

411 Ops

146 Paths



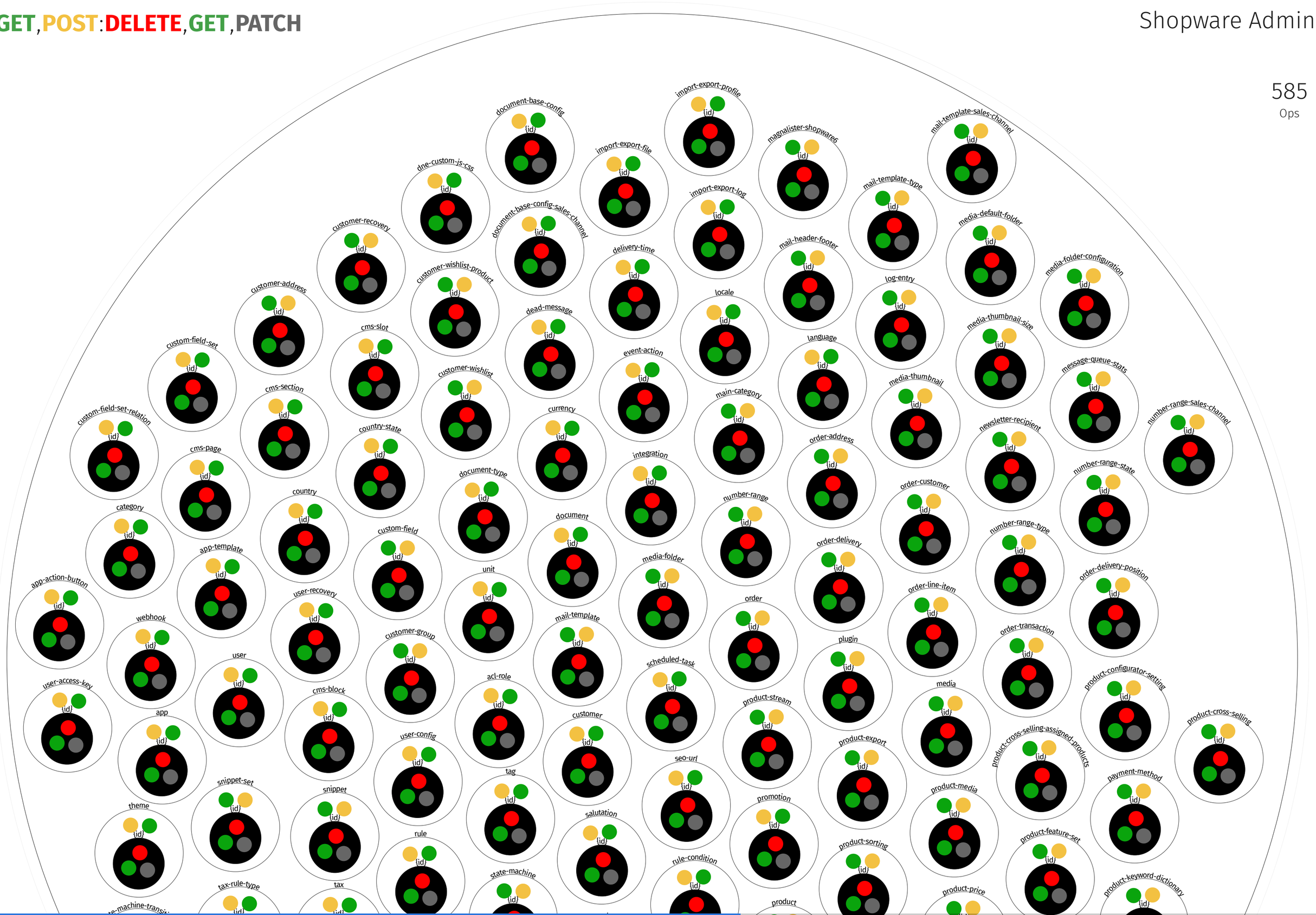
GET,POST:DELETE,GET,PATCH

Shopware Admin API

3

585 Ops

234 Paths



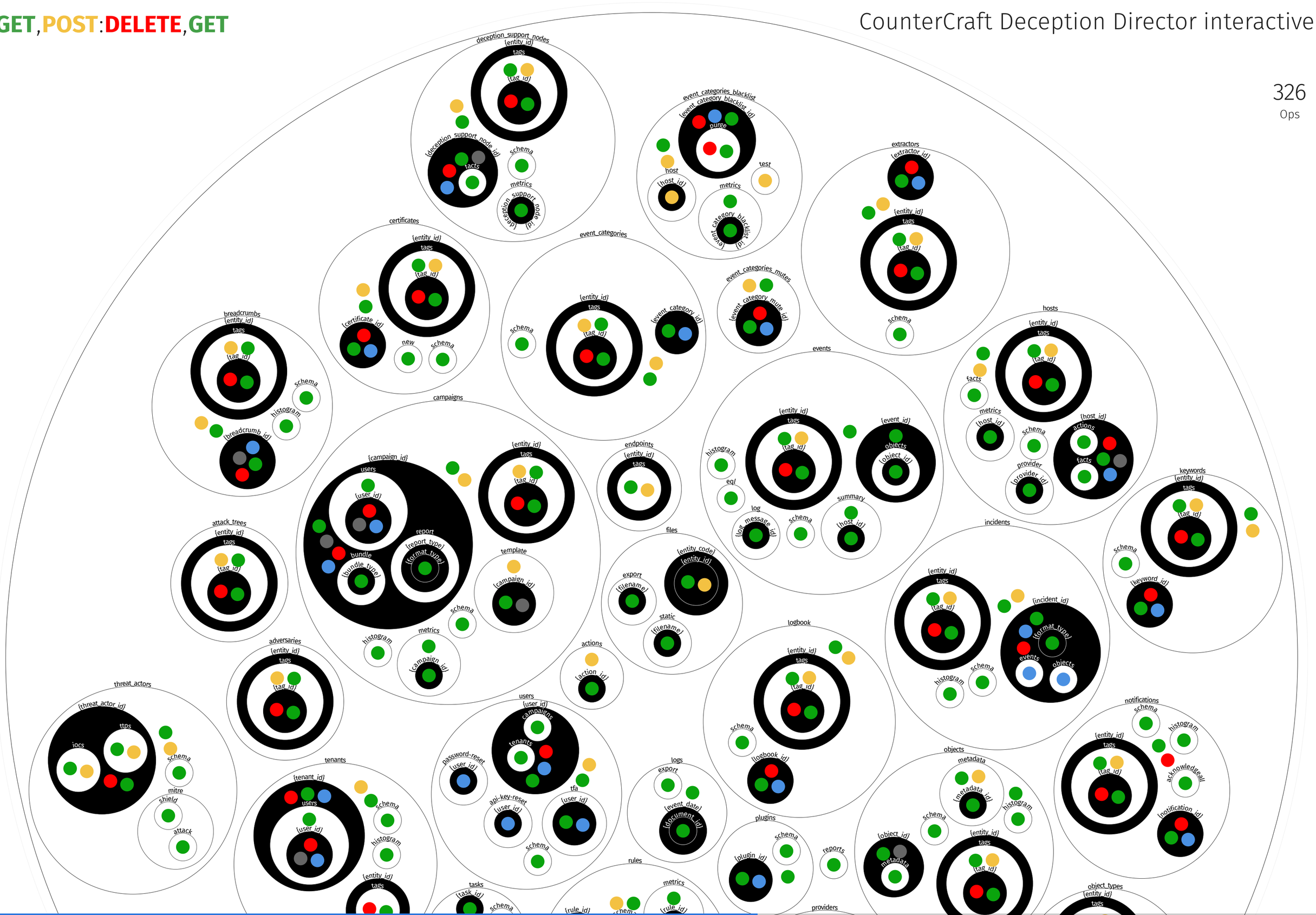
GET,POST:DELETE,GET

CounterCraft Deception Director interactive API

1

326 Ops

196 Paths



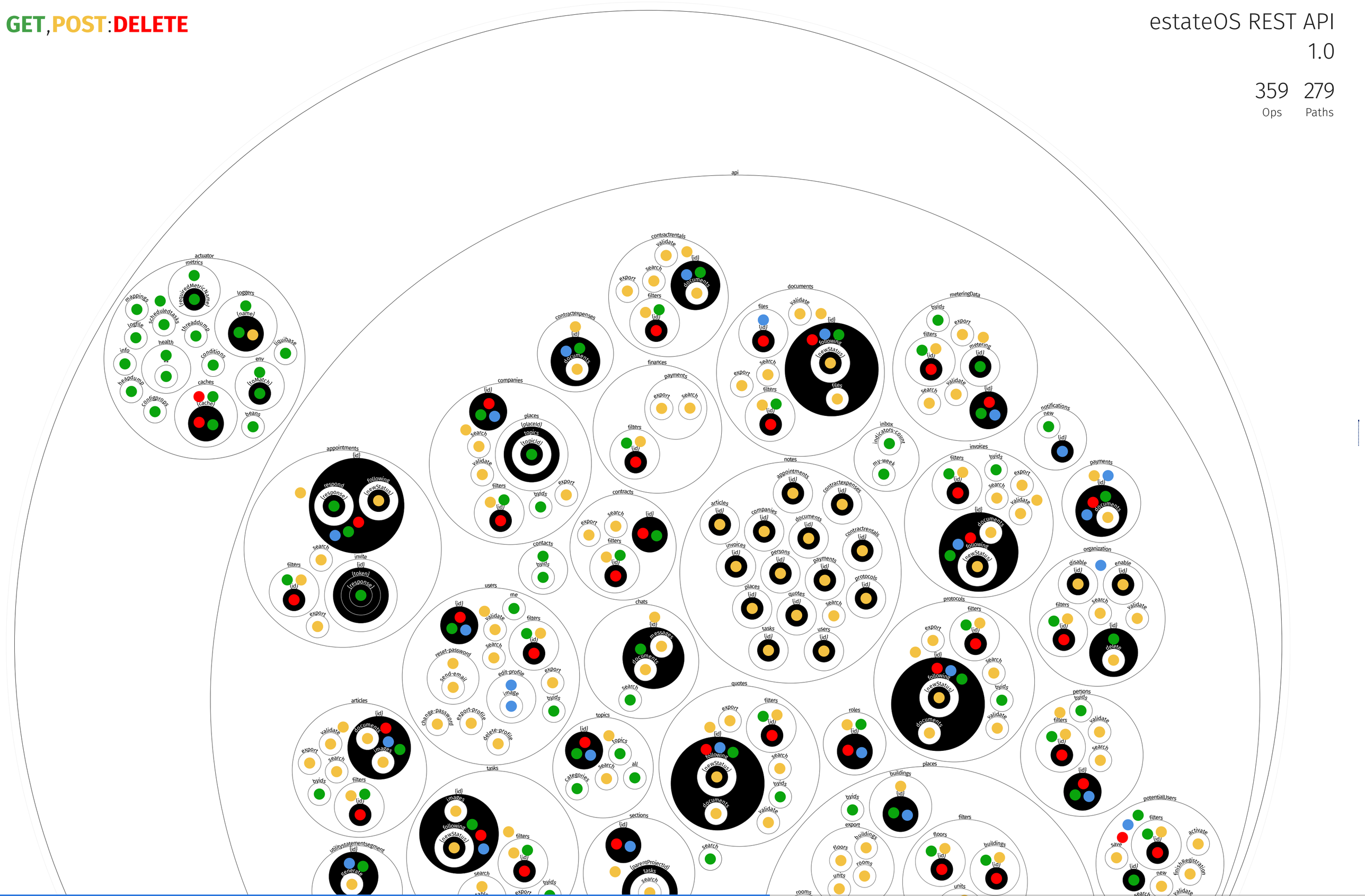
GET,POST:DELETE

estateOS REST API

1.0

359 Ops

279 Paths



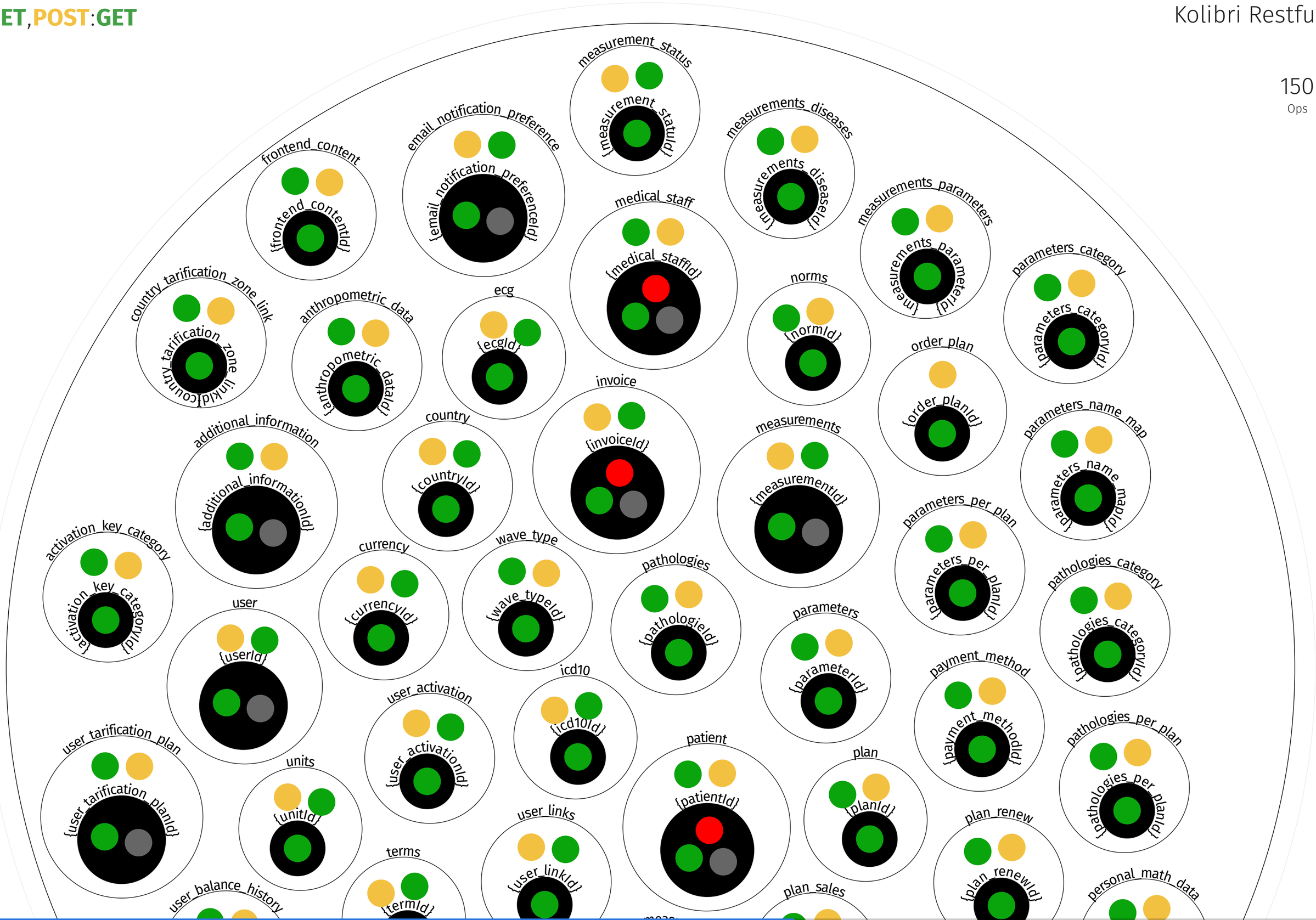
GET,POST:GET

Kolibri Restful API

1.0

150 Ops

92 Paths



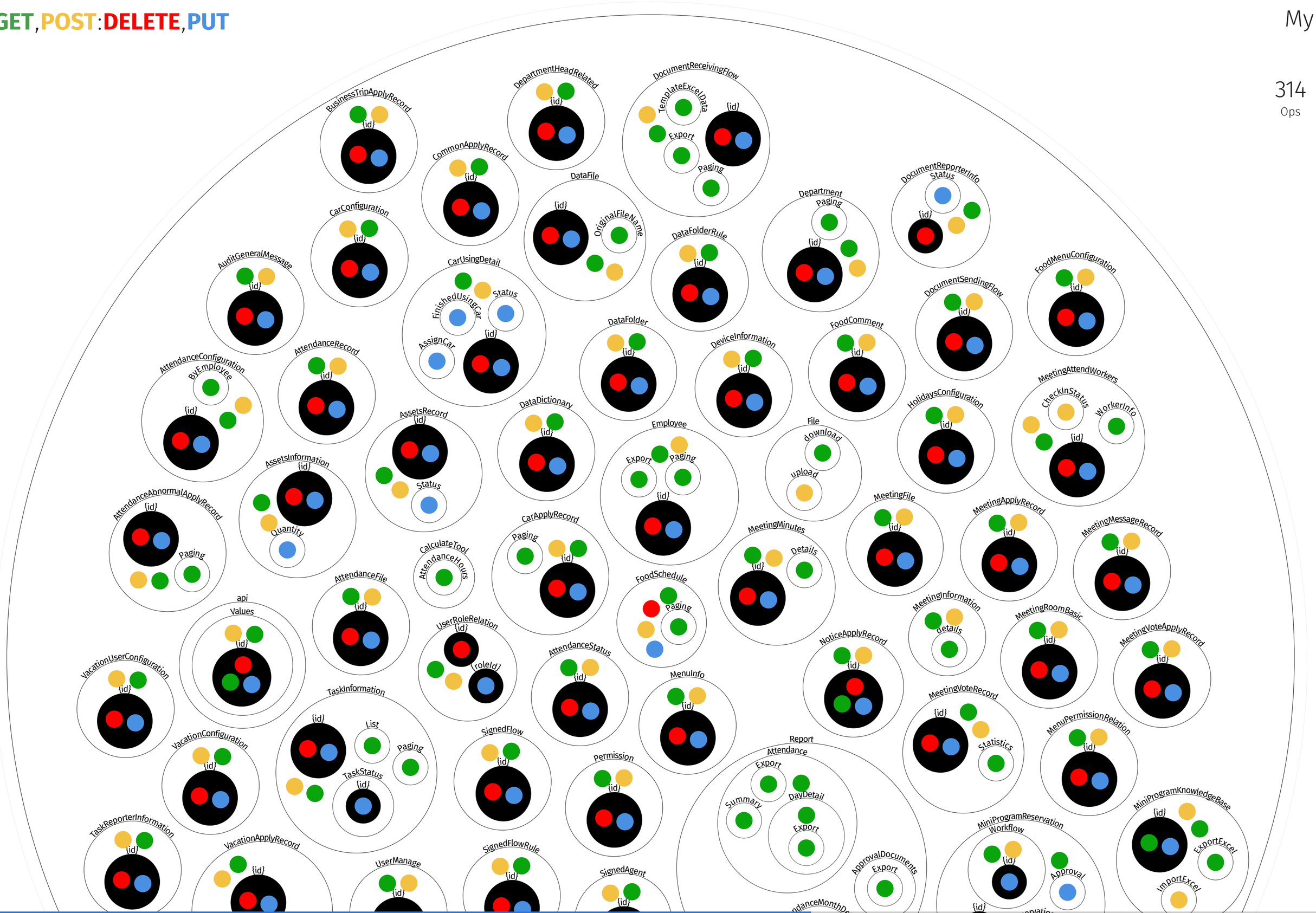
GET,POST:DELETE,PUT

My API

v1

314 Ops

185 Paths



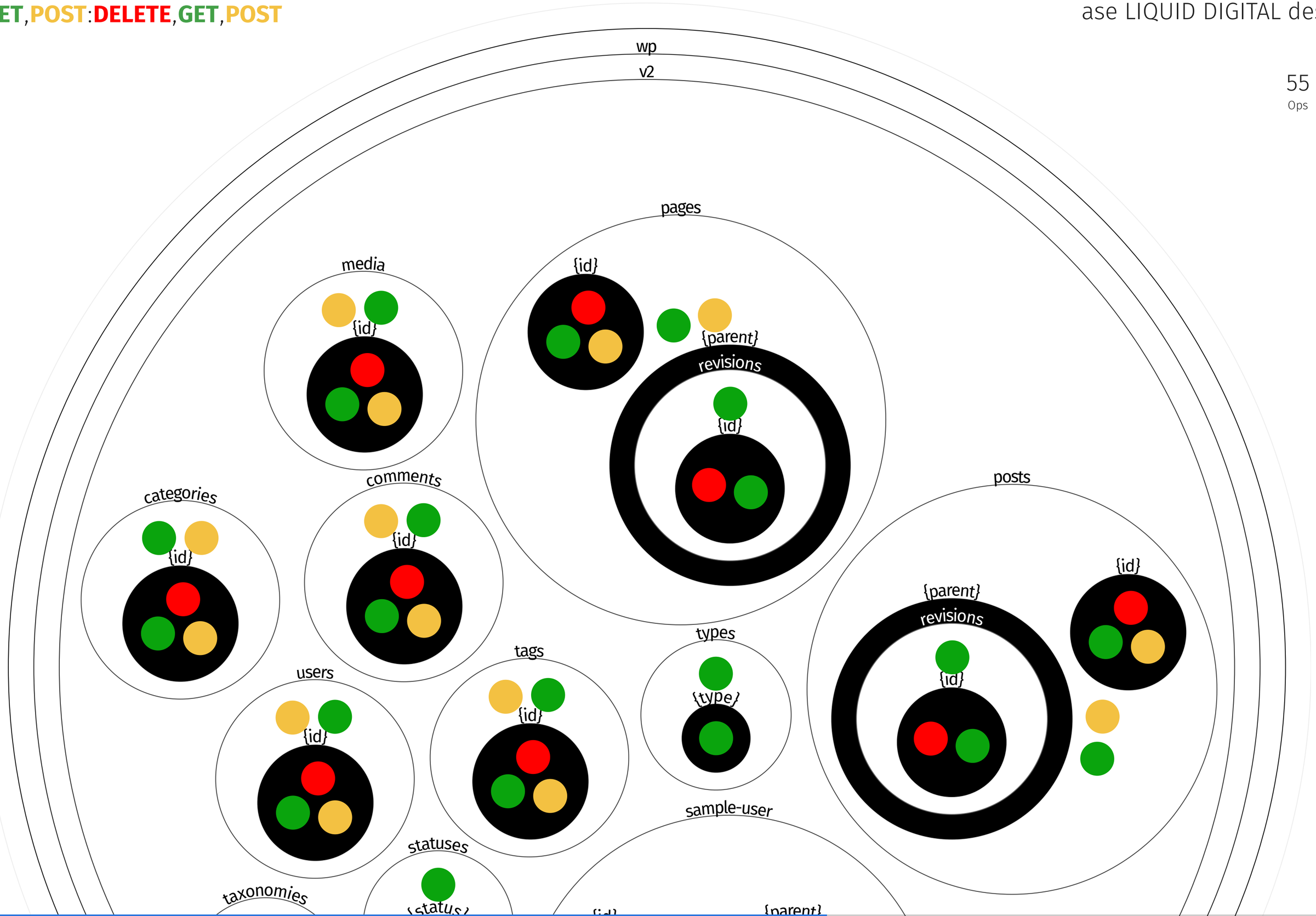
GET,POST:DELETE,GET,POST

ase LIQUID DIGITAL design

1.0

55 Ops

28 Paths



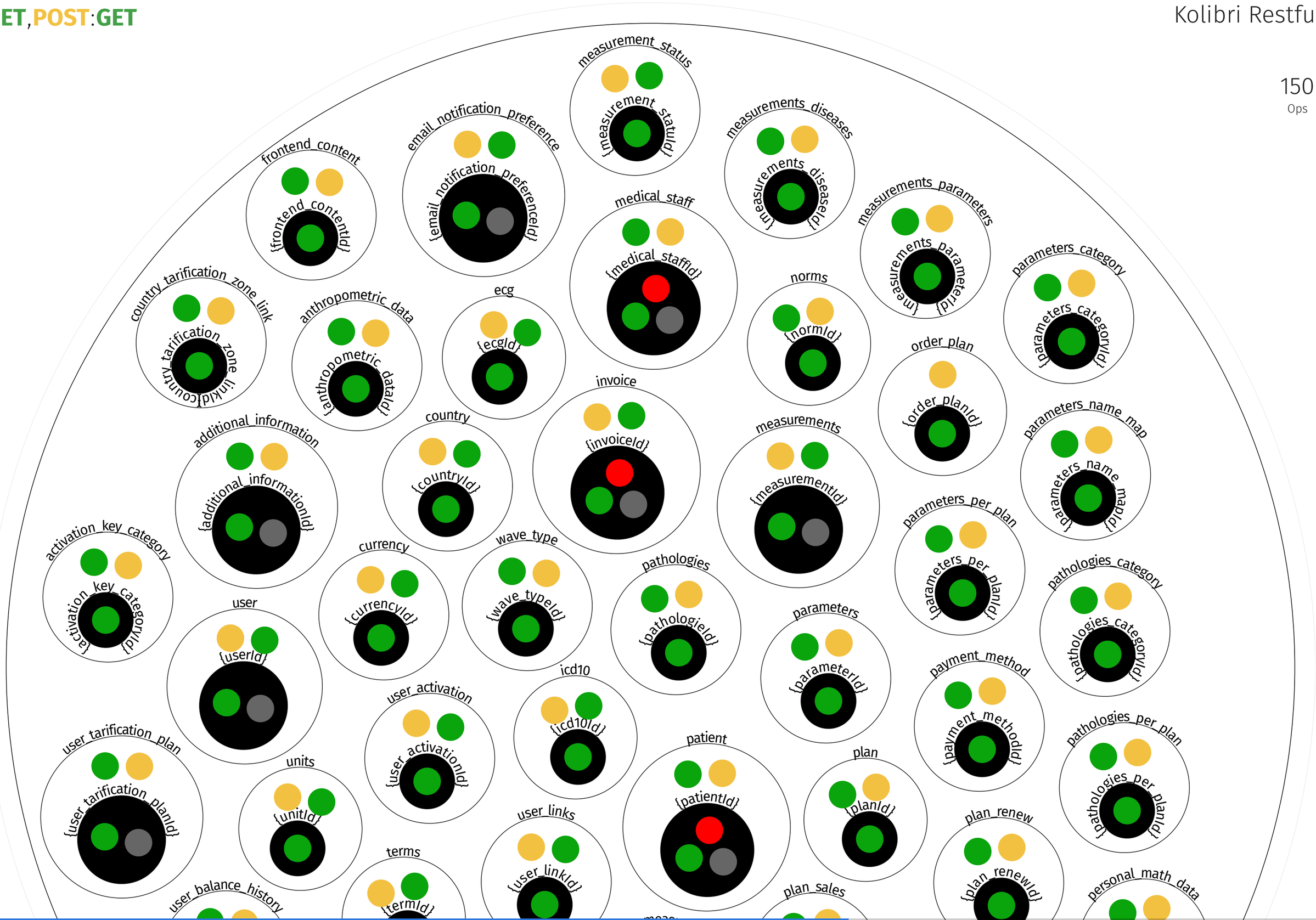
GET,POST:GET

Kolibri Restful API

1.0

150 Ops

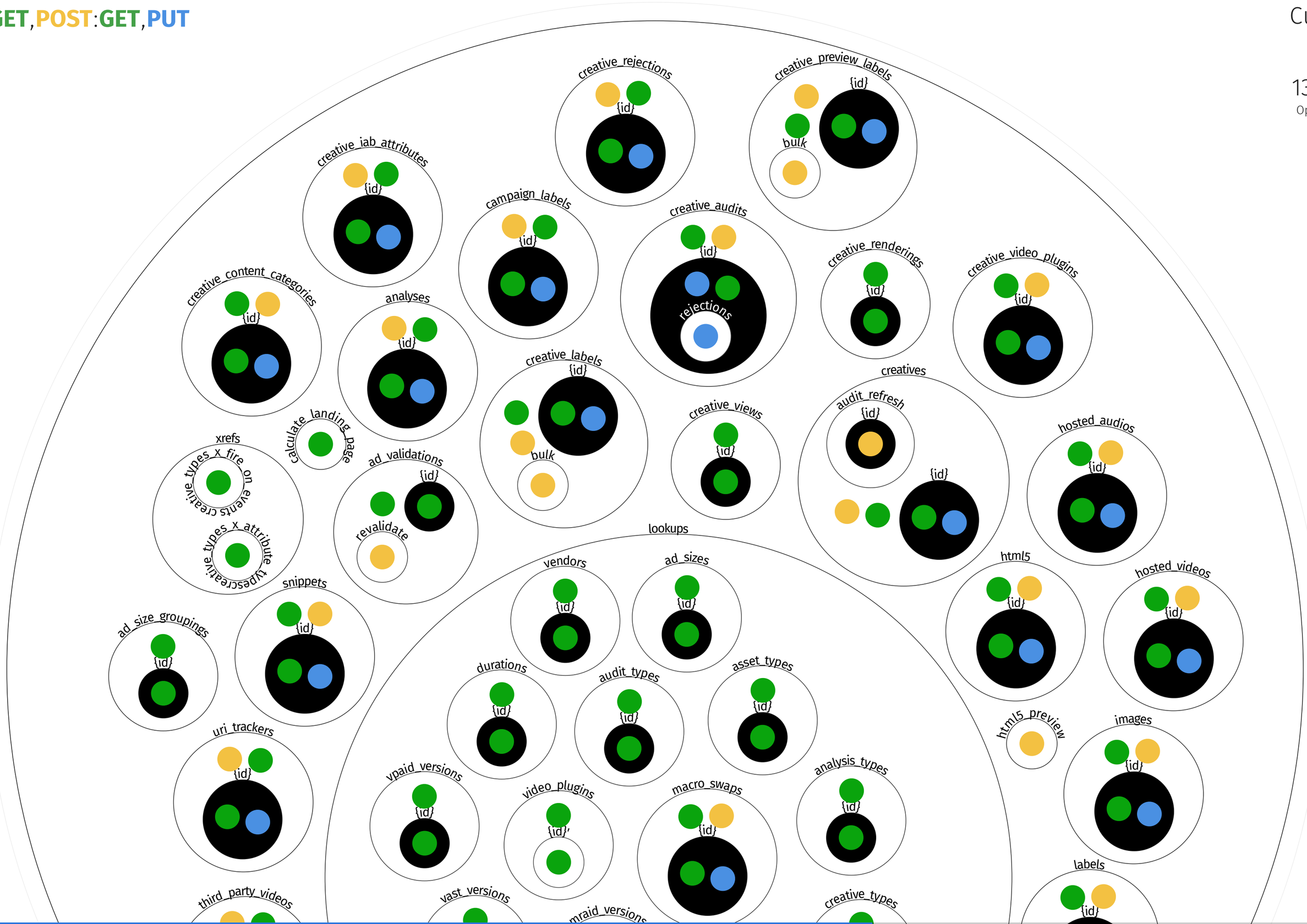
92 Paths



GET,POST:GET,PUT

Curator
2.4.0

131 Ops
89 Paths



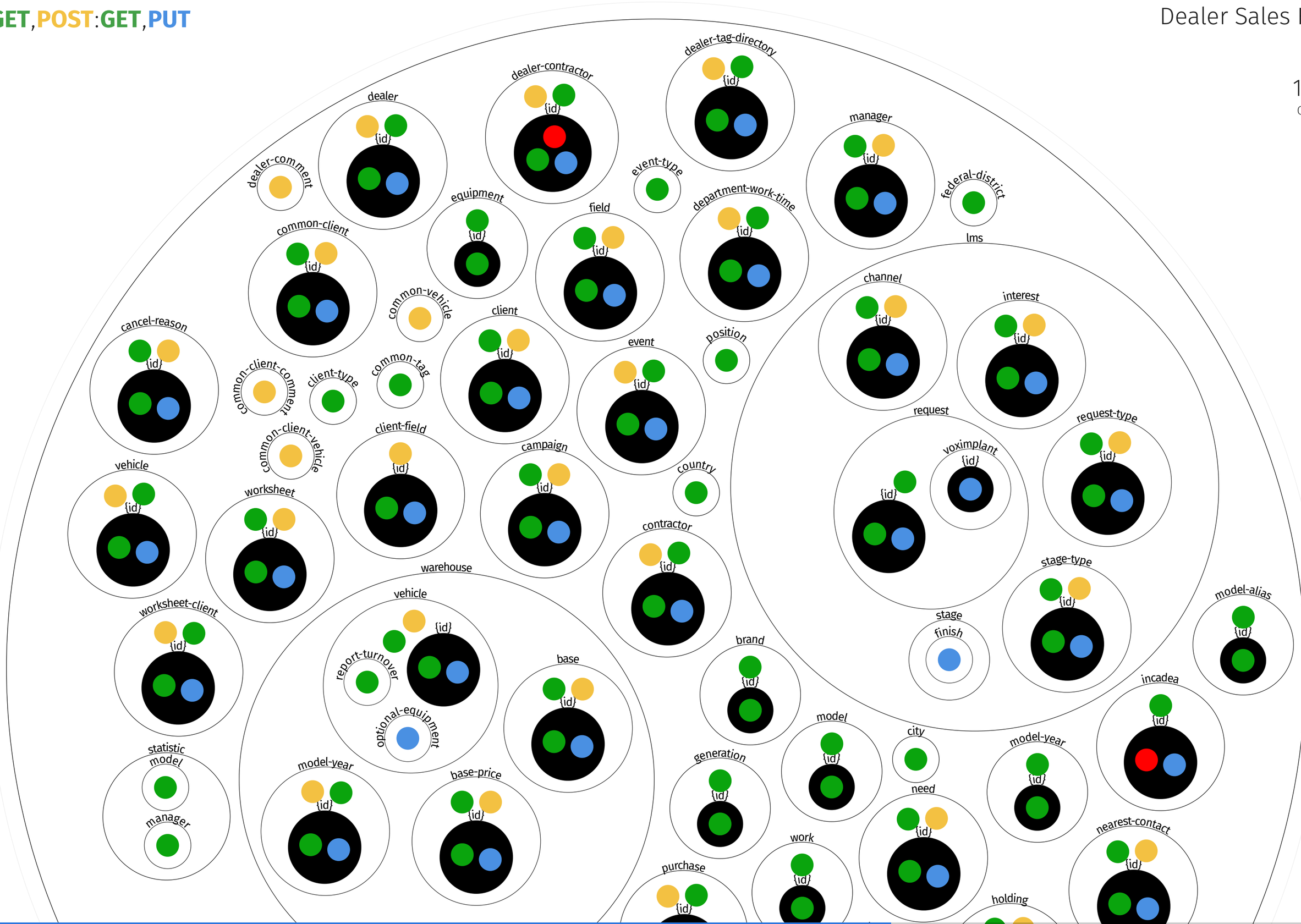
GET,POST:GET,PUT

Dealer Sales Funnel

0.0.1

177 Ops

113 Paths



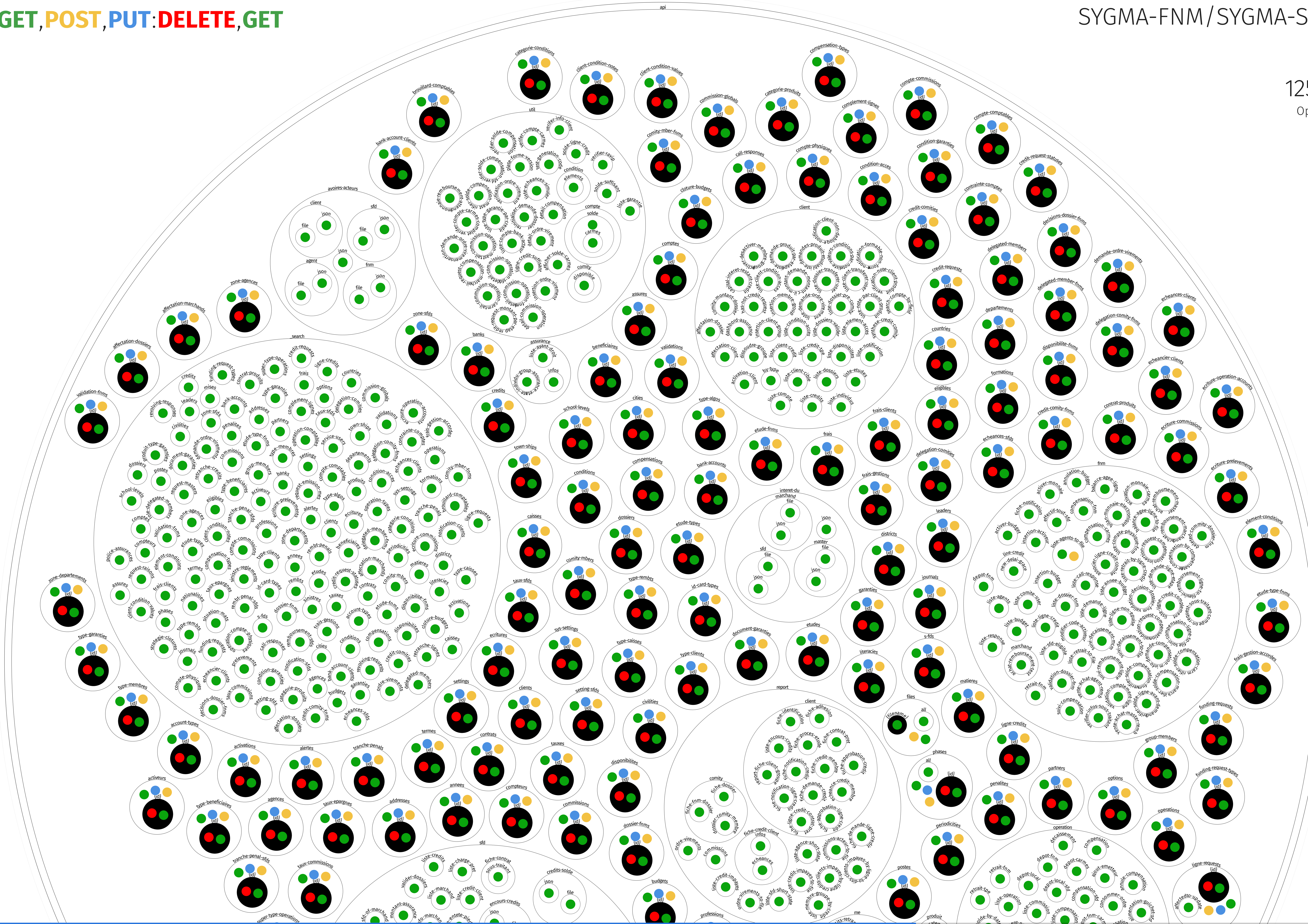
GET,POST,PUT:DELETE,GET

SYGMA-FNM/SYGMA-SFD API

0.0.1

1252 Ops

782 Paths



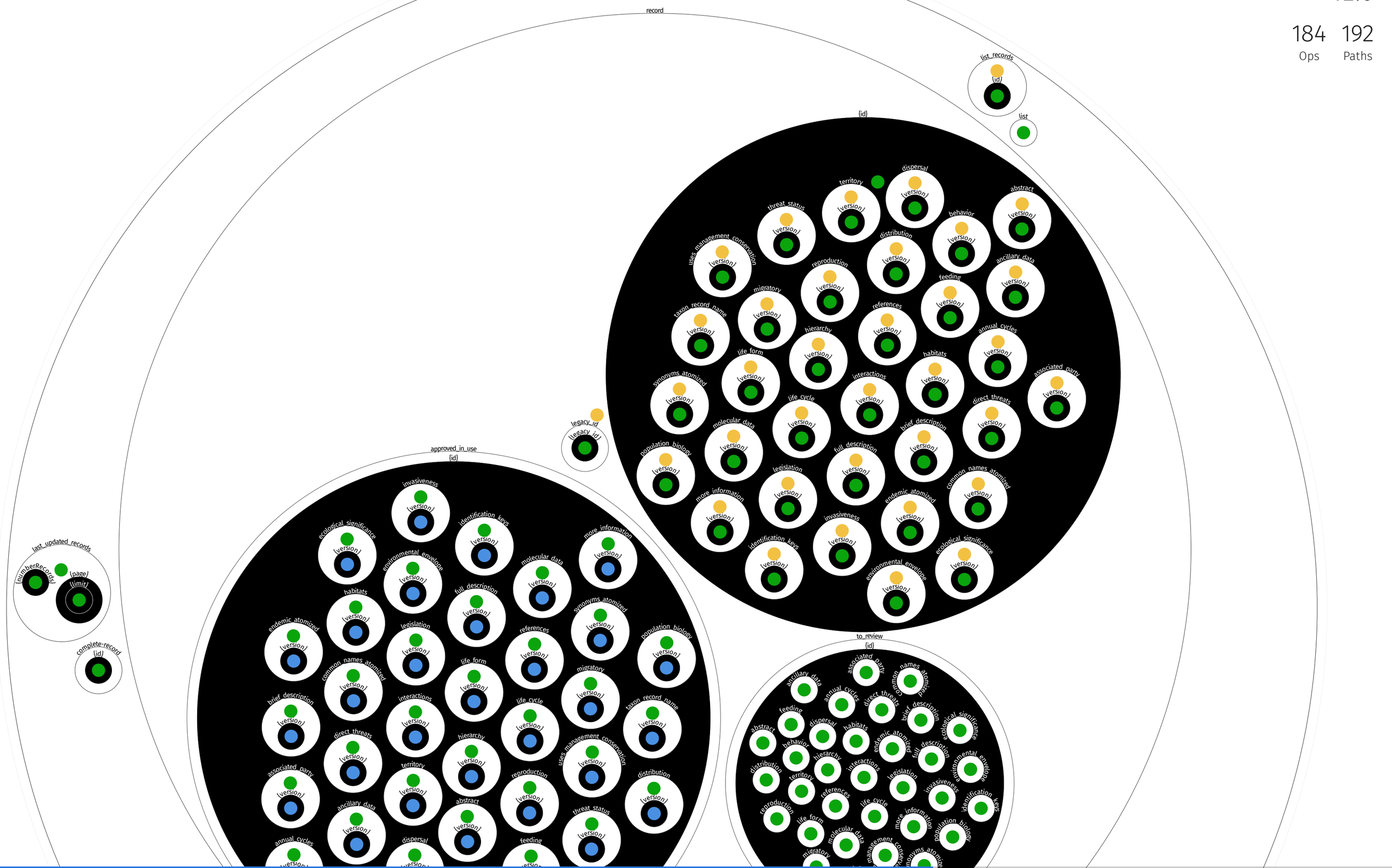
GET:PUT

POST:GET

Catalogue API v2.0

184 Ops

192 Paths



GET:DELETE,GET,PUT

NetworkManagementClient
2016-06-01

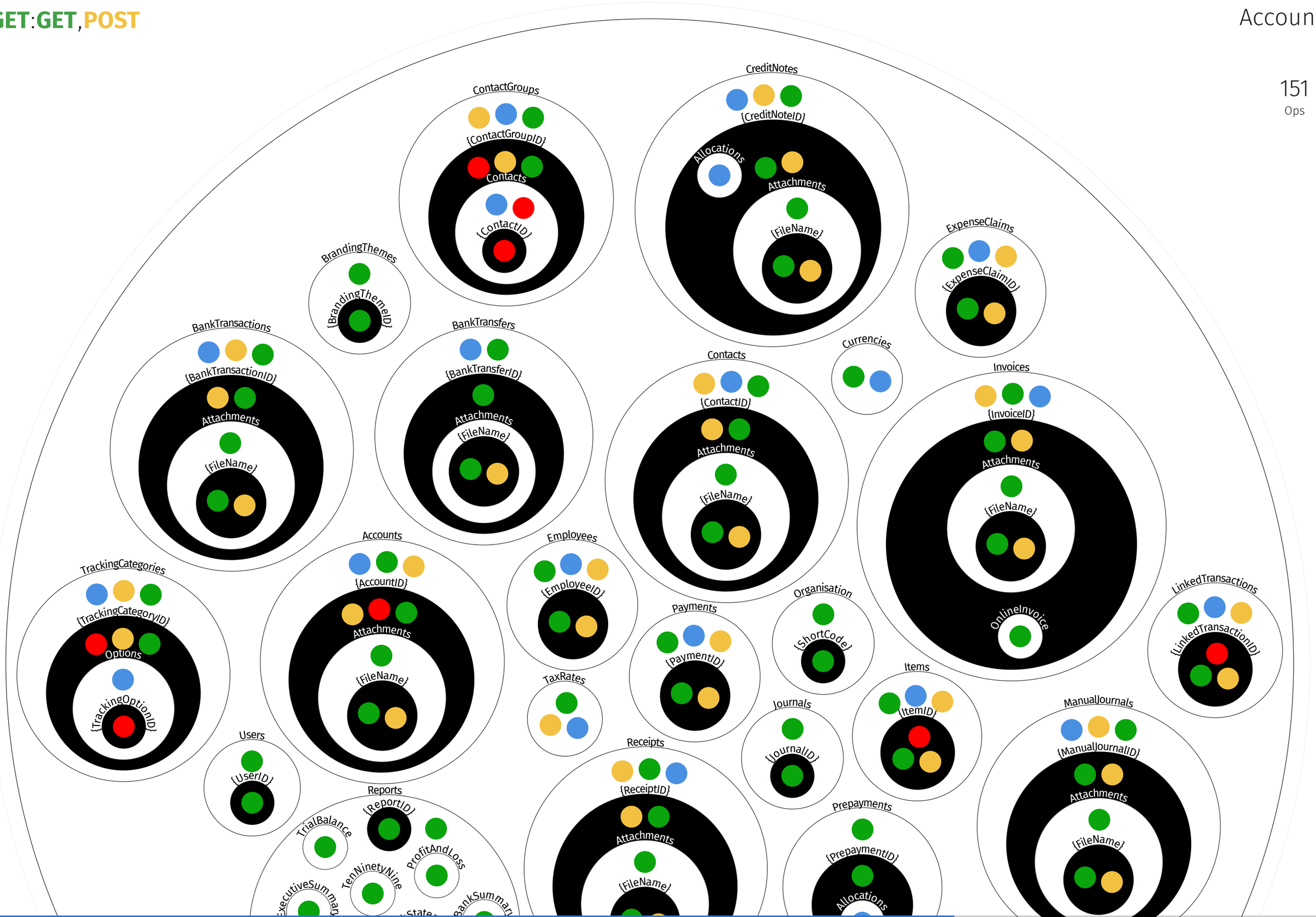
97 Ops
62 Paths

GET:**GET**,**POST**

Accounting 2.0

151 Ops

87 Paths



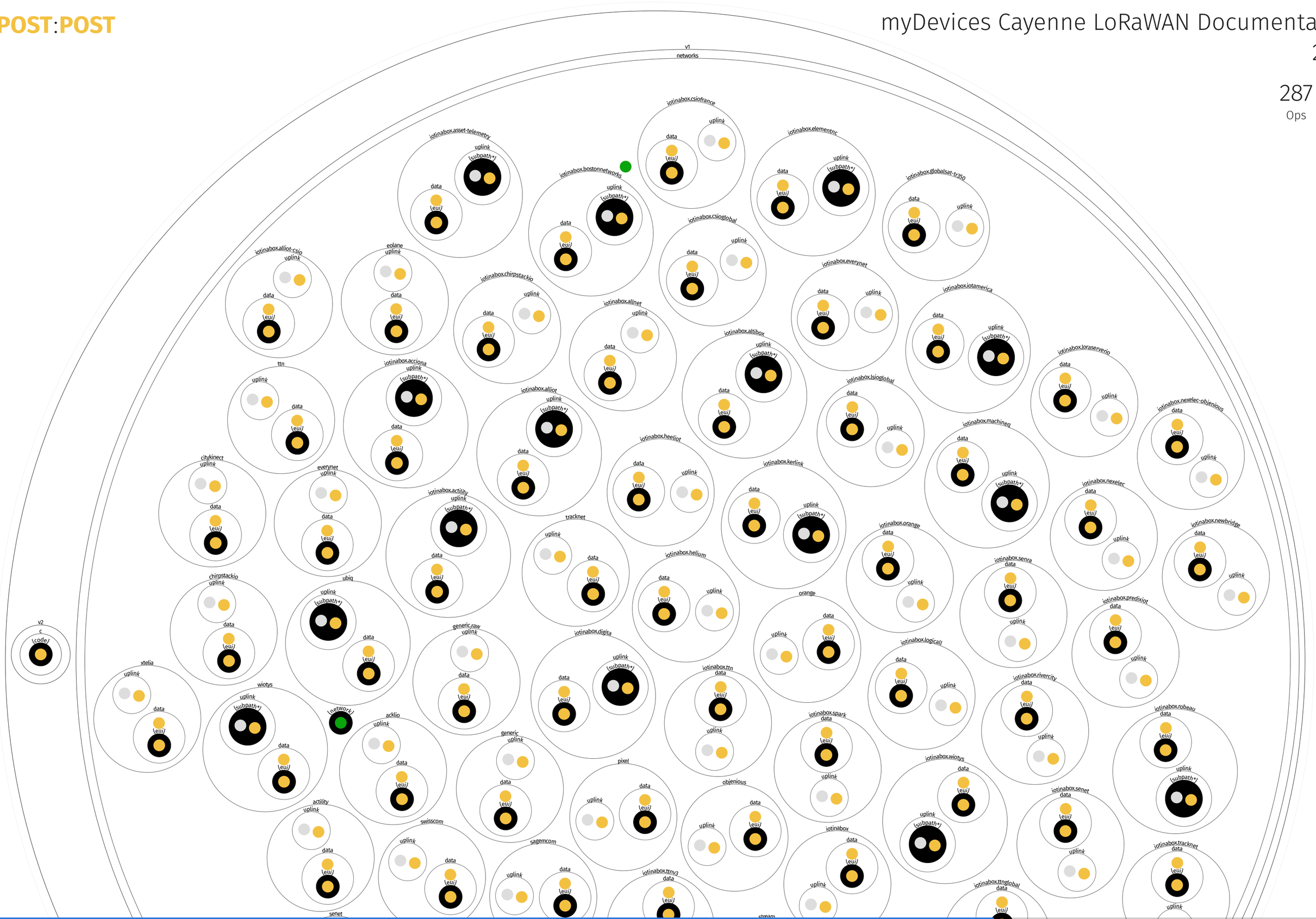
POST:POST

myDevices Cayenne LoRaWAN Documentation

2.2.0

287 Ops

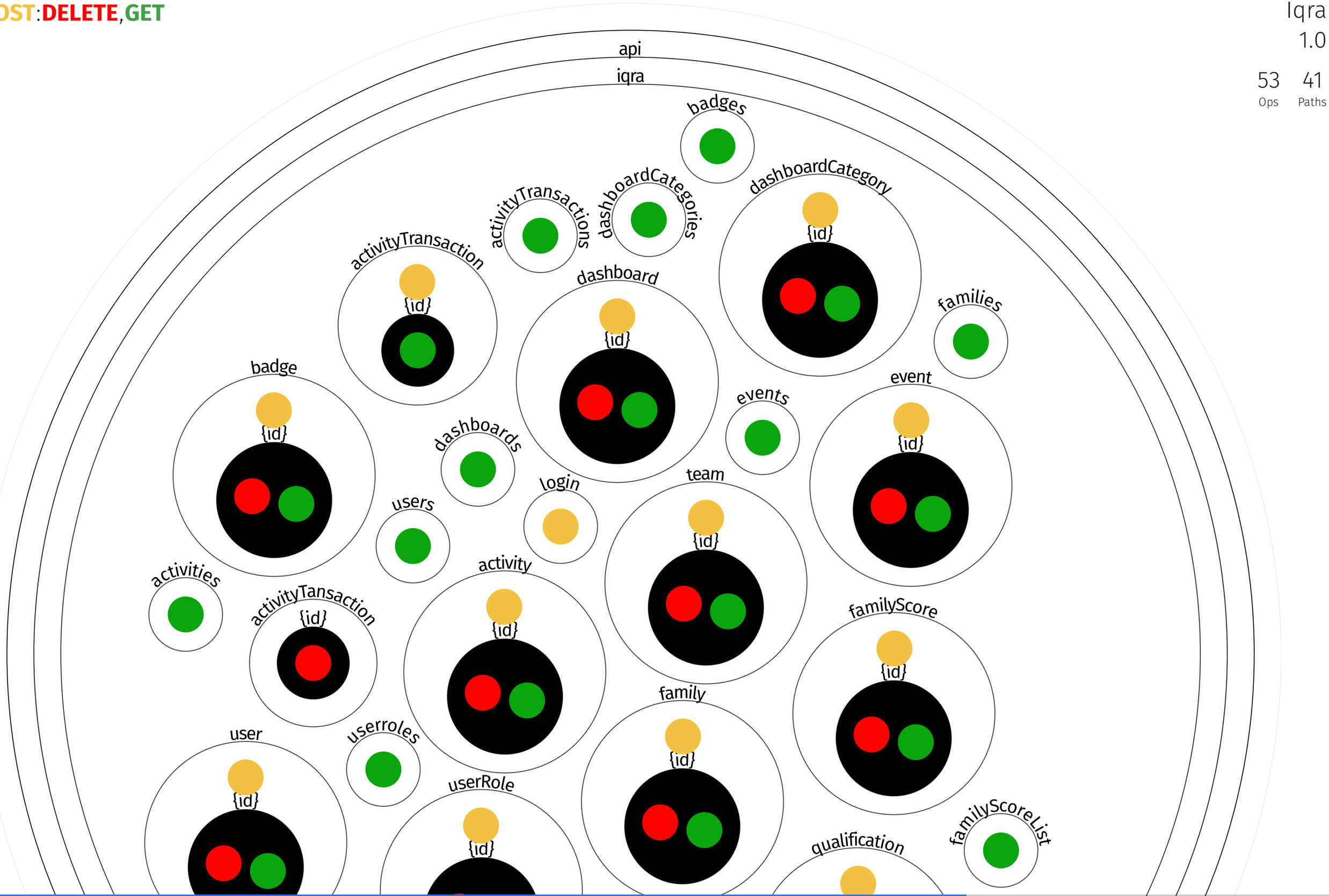
216 Paths



POST:DELETE,GET

Iqra
1.0

53 Ops
41 Paths



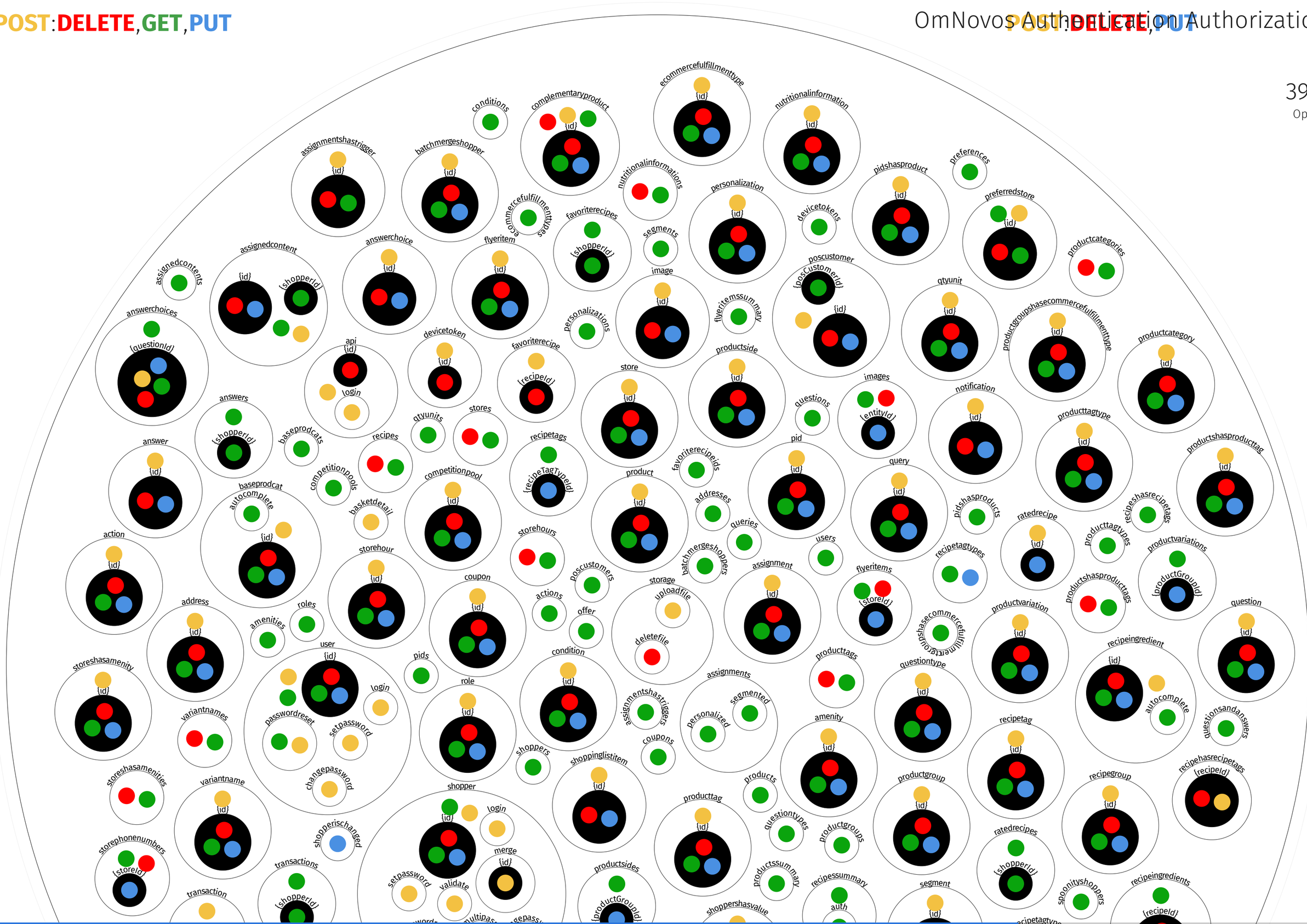
POST:DELETE,GET,PUT

OmNovos Authentication Authorization API

0.0.1

393 Ops

252 Paths



Version Identifier

How can an API provider indicate its current capabilities as well as the existence of possibly incompatible changes to clients, in order to prevent malfunctioning of clients due to undiscovered interpretation errors?

Version Identifier

Metadata

```
1.0 1.0.0 v1 0.0.1 undefined version not set 2.0 0.1 v2
0.1.0 1.0.1 2.0.0 1.1.0 v1.0 3.0 0.1.0-SNAPSHOT 0.0.1-
SNAPSHOT beta 3.0.0 1.1 1.0.0-SNAPSHOT 3.0.2 1.0.2
1.2.0 unversioned v3 0.0.0 0.2.0 2015-11-01 v1.0.0 V1
v0.11 2018-06-01-preview 1.0-SNAPSHOT 2019-08-01 v1.7.0
版本号:1.0.0 3.0.1 2019-07-01 V1.0 0.0.2 2019-06-01
2019-01-01 API V1.0 v0.1 2016-05-01 4.0.0 master 2014-
04-01 2018-02-01 2019-04-01 1.0.5 v1.8.0 2018-01-01
2017-10-01 2019-12-01-preview 1.0.3 1.3.0 v0 v1beta1
0.0.3 0.1.1 2018-06-01 版本号:2.3.0 v1.6.0 2018-10-01
2017-03-01-preview 1.0.4 1.4.0 M1 0.2 1.1.1 2.0.1 版本
号:1.0 1.5 V0.0.1 v0.0.1 2017-03-01 2018-07-01 1.1.3
```

Version Identifier

First Path Segment

```
v1 v2 v1.0 v3 v4 v2.0 v9 v8 v0 v1.1 v0.1 v{version} V1
v{VERSION_NBR} v2.1 v1.2 v0.2 v{apiVersion} V1.0 v5
v1.3 v{ver} V3 v001 {v} v300 V2 v{api-version} v7 v3.0
v0.3 v2020 favicon_v{v}.ico v6 v20180820 v20190125 v01
v11 {api.v1.prefix} v1.4 v3.5 v{1} V4 v1.5 v2.2 v1.6
v3.3 v0.4 v0.11 v20 v0.9 v1.{output_format} v7.0 v03 V2
V5 V6 v2 v1{image_folder} v360 v20170314 V20161128
v20160101 V20170117 V20190624 v20180301 V3 v20180227
V20150921 V2016 V20180706 V2015 v20140515 v20131101
v1.56 V20141113
```

Two in Production

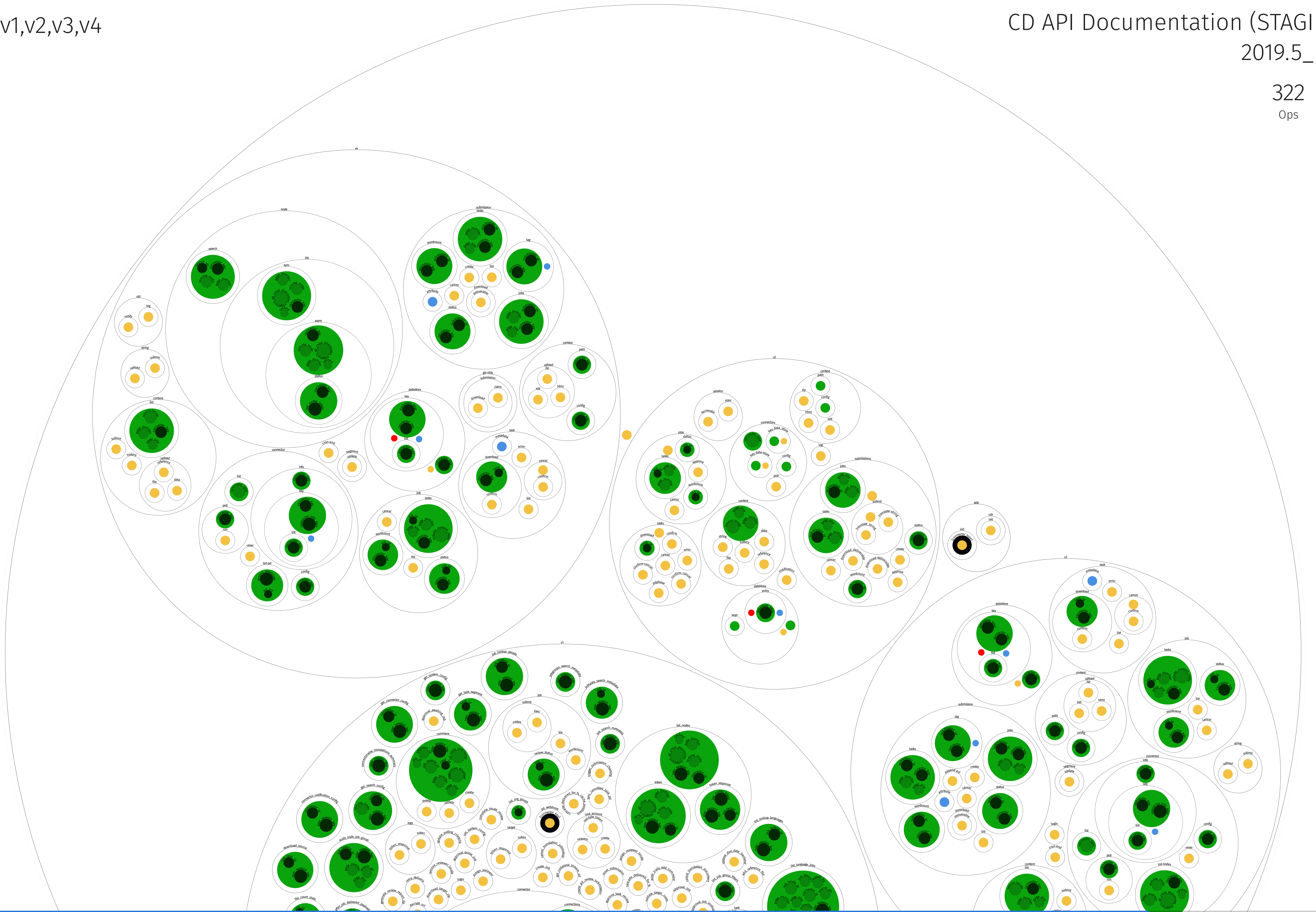
How can a provider gradually update an API without breaking existing clients, but also without having to maintain a large number of API versions in production?

v1,v2,v3,v4

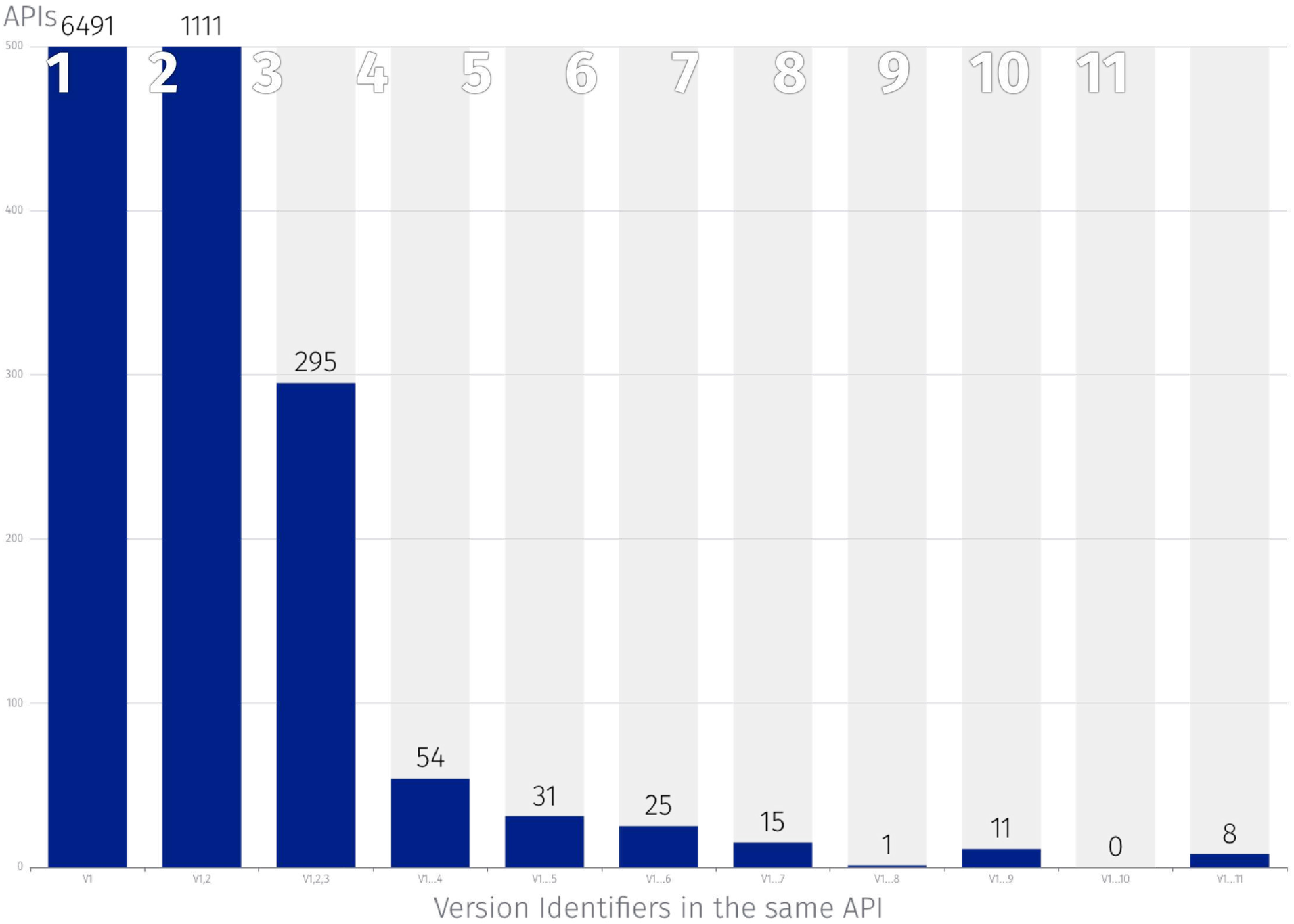
CD API Documentation (STAGING)

2019.5_RC2

322 Ops | 307 Paths



N in Production



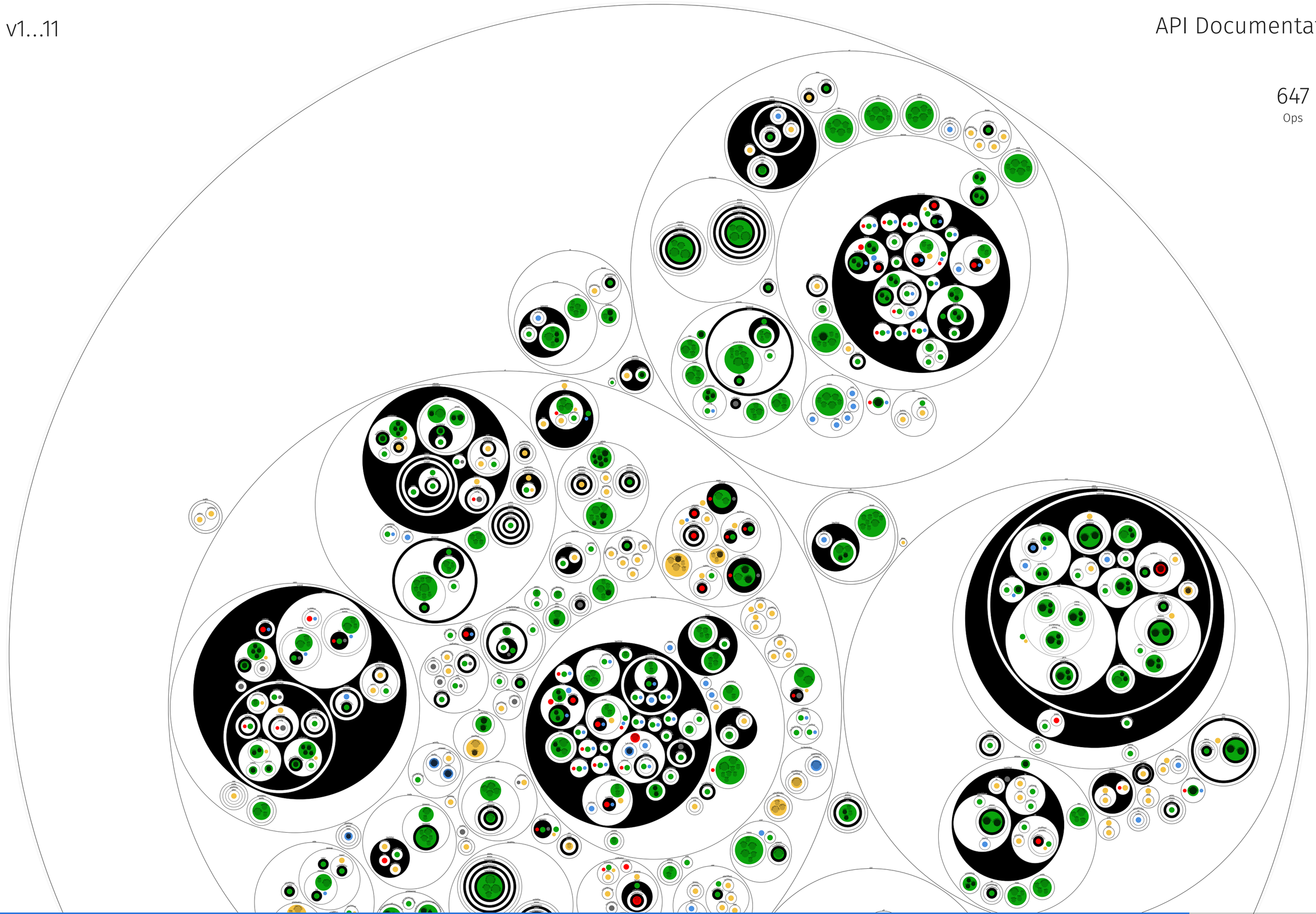
v1...11

API Documentation

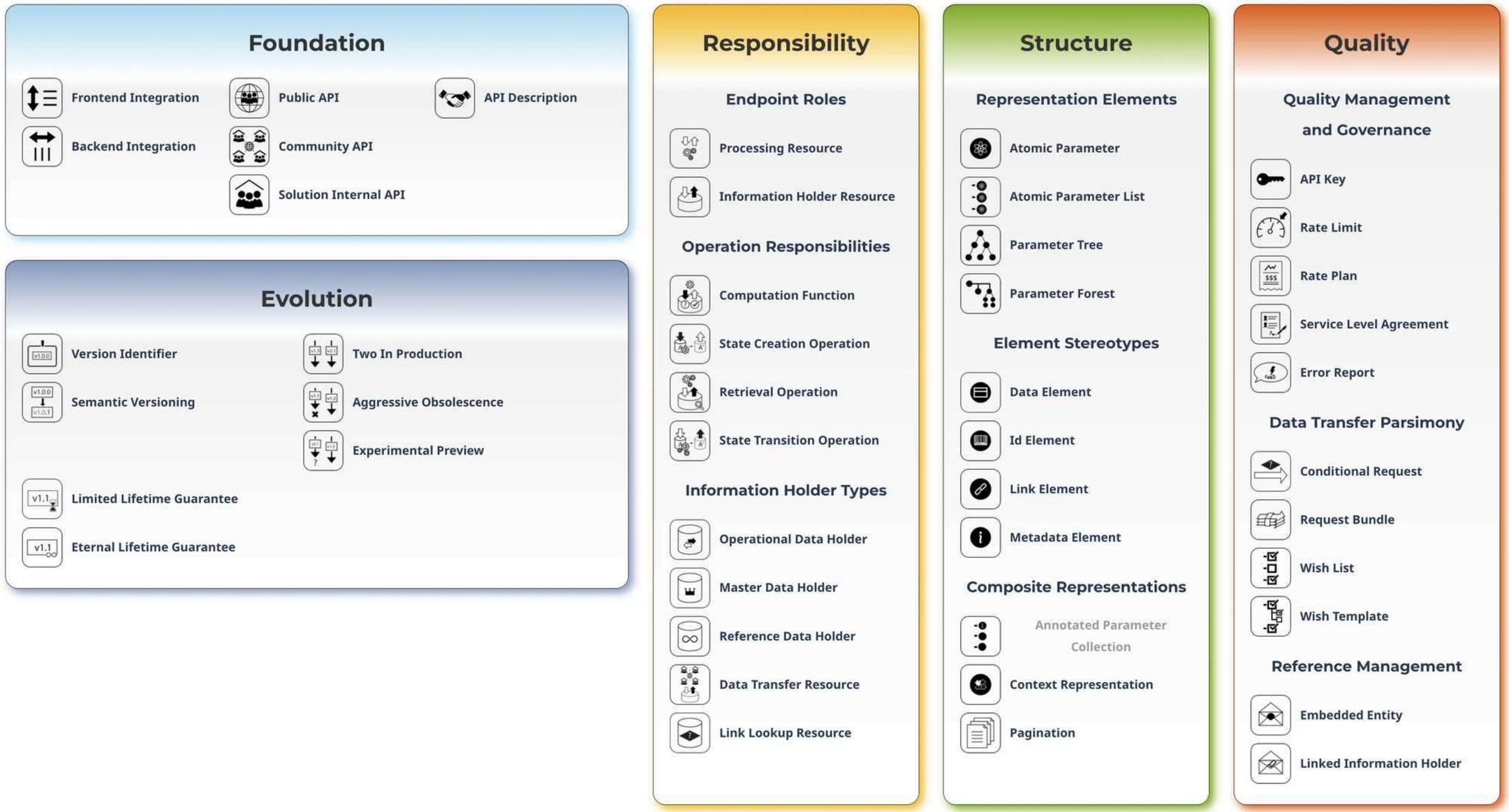
1.1.0

647 Ops

519 Paths



Microservice API Design Patterns



https://microservice-api-patterns.org

The Microservice Hypothesis

By keeping them small, microservice architectures can foster their autonomy and independent evolution.

Small ⇒ Loosely Coupled

Loosely Coupled ⇒ Small

Acknowledgements

Souhaila Serbout, Fabio Di Lauro, Segilola Mustapha, Gustavo Graziani, Albert Walser, Alessandro Romanelli, Ana Ivanchikj, Apitchaka Singjai

Olaf Zimmermann, Uwe Zdun, Daniel Lübke, Mirko Stocker

Erik Wilde, Shubham Shah (Assetnote)

References

- Souhaila Serbout, Fabio Di Lauro, Cesare Pautasso, **Web APIs Structures and Data Models Analysis**, 19th IEEE International Conference on Software Architecture (ICSA 2022), Honululu, Hawaii, IEEE, March, 2022.
- Souhaila Serbout, Cesare Pautasso, Uwe Zdun, Olaf Zimmermann, **From OpenAPI Fragments to API Pattern Primitives and Smells**, Proc. of the European Conference on Pattern Languages of Programs (EuroPLoP 2021), Kloster Irsee, Germany, July 2021
- Fabio Di Lauro, Souhaila Serbout, Cesare Pautasso, **Towards Large-scale Empirical Assessment of Web APIs Evolution**, Proc. 21st International Conference on Web Engineering (ICWE2021), Biarritz, France, Springer, May 2021 (Best Paper Award)
- Cesare Pautasso, Ana Ivanchikj, Silvia Schreier, **A Pattern Language for RESTful Conversations**, Proc. of the 21st European Conference on Pattern Languages of Programs (EuroPLoP 2016), Kloster Irsee, Germany, July 2016, pp. 4:1-4:22
- Daniel Lübke, Olaf Zimmermann, Cesare Pautasso, Uwe Zdun, Mirko Stocker, **Interface Evolution Patterns - Balancing Compatibility and Flexibility across Microservices Lifecycles**, Proc. of the 24th European Conference on Pattern Languages of Programs (EuroPLoP 2019), Irsee, Germany, July 2019
- Olaf Zimmermann, Mirko Stocker, Uwe Zdun, Daniel Lübke, Cesare Pautasso, **Introduction to Microservice API Patterns (MAP)**, Joint Post-proceedings of the First and Second International Conference on Microservices (Microservices 2017/2019)
- Cesare Pautasso, **Beautiful APIs**, Leanpub, 2020
- Cesare Pautasso, **Beautiful API Evolution**, Leanpub, 2021
- Cesare Pautasso, **Beautiful BIG APIs**, Leanpub, 2022