

Mojolicious and Nuclino

About me

Renée Bäcker

RENEEB @
Github
CPAN

Perl-Services.de since 2011

Mojolicious

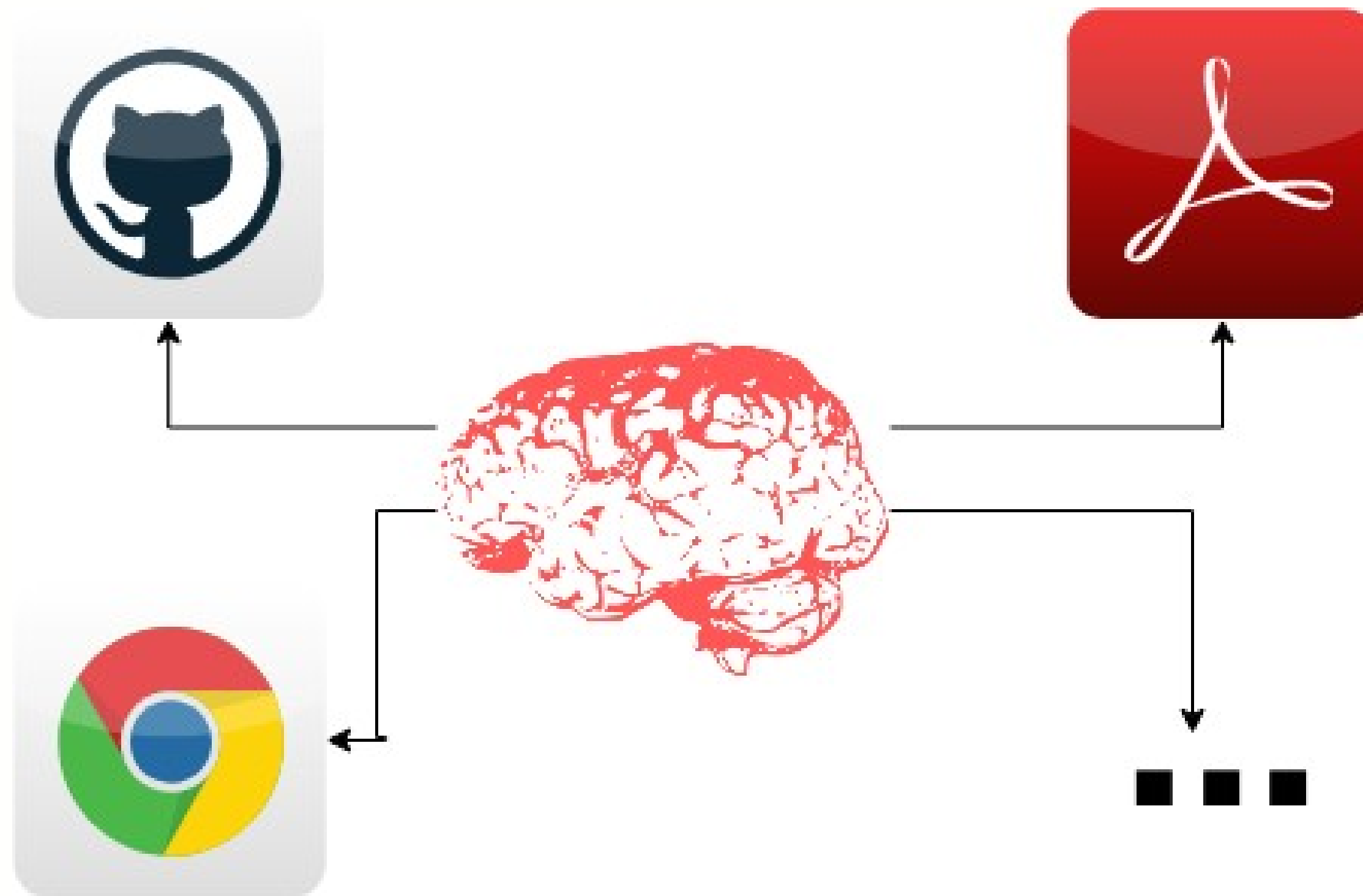
Nuclino

Our „Single Source of Truth“:

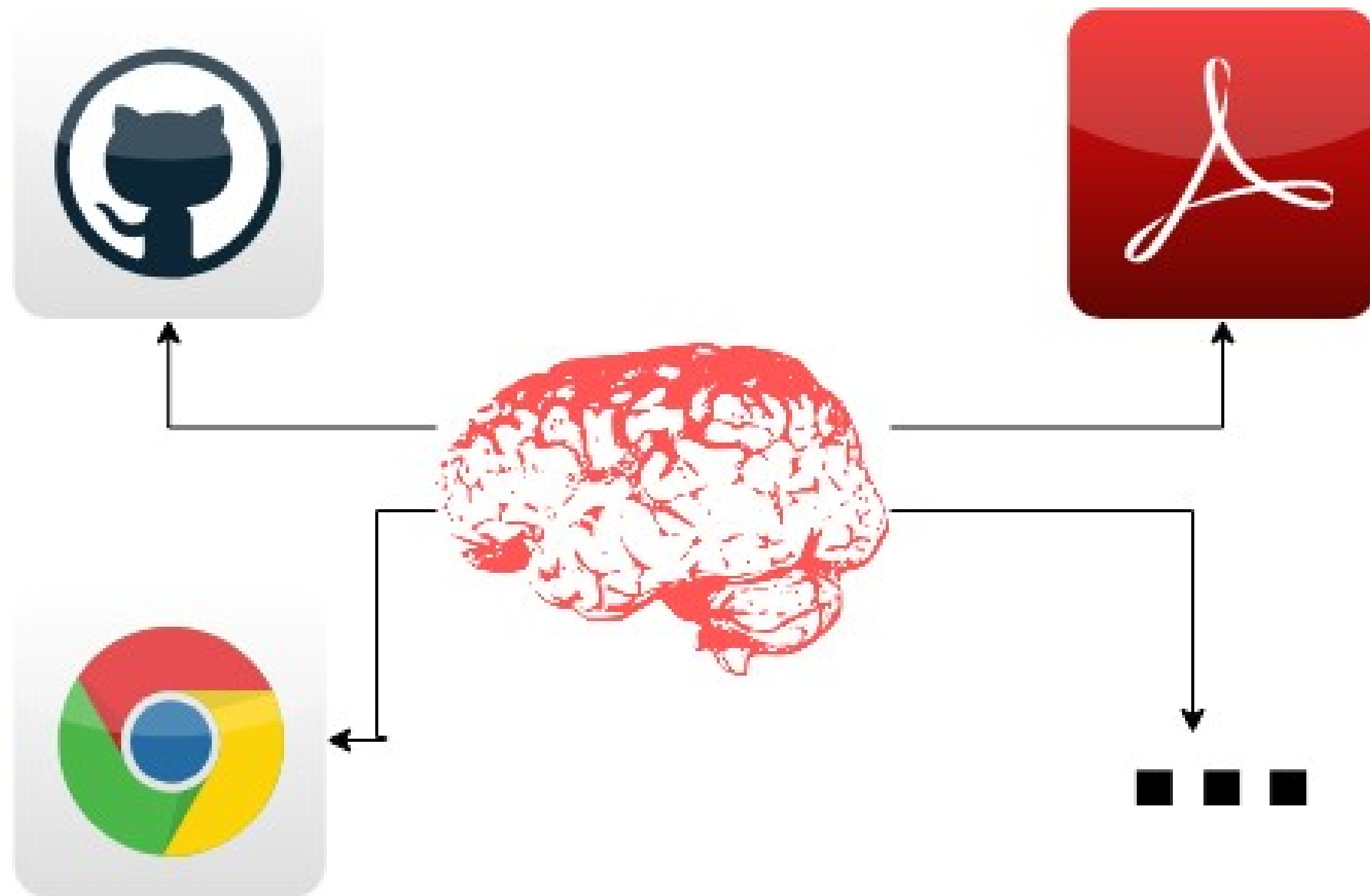
- Project documentation
- Internal workflows
- Plannings
- Manuals
- ...

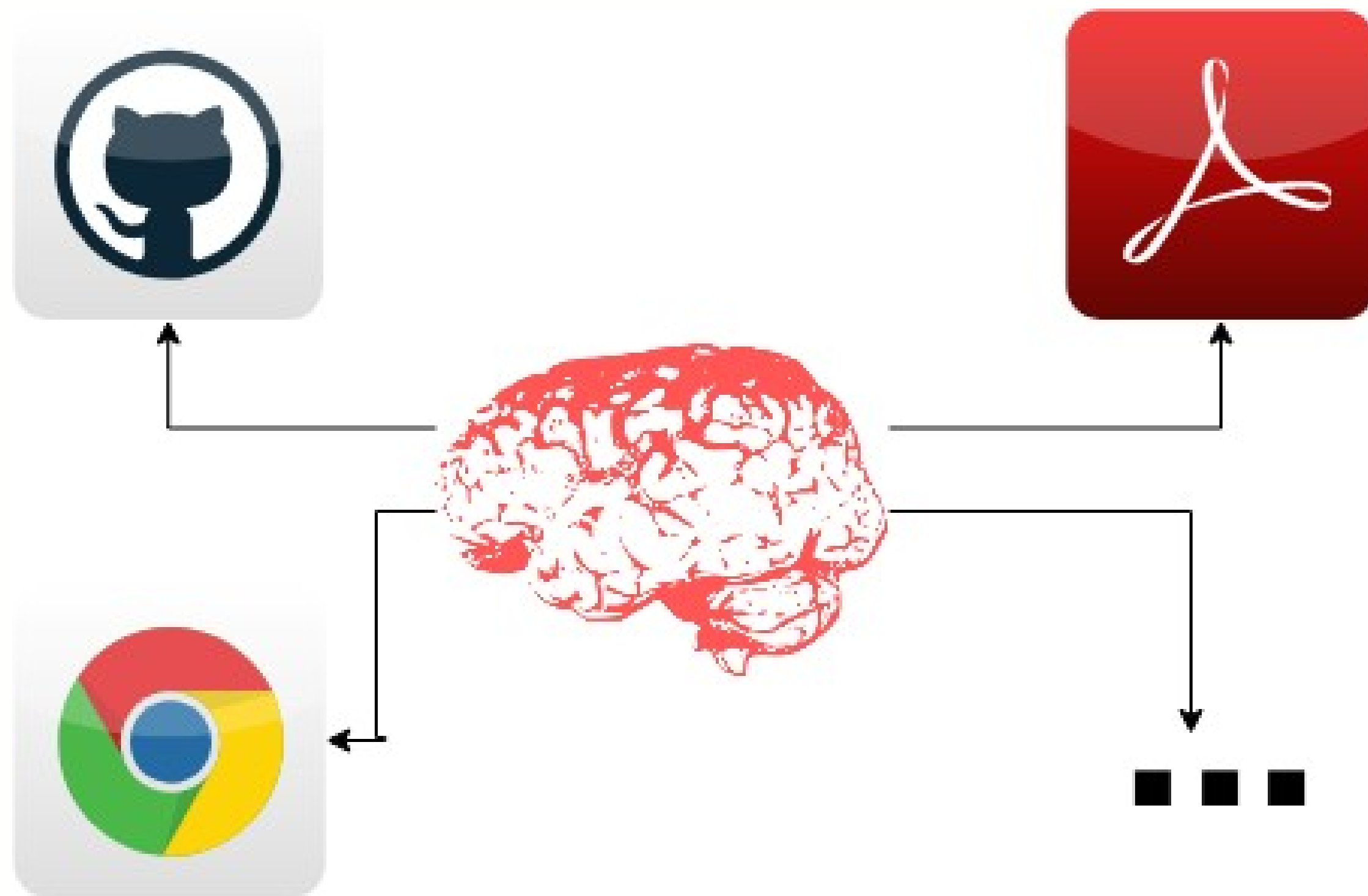
Video: <https://www.perl-academy.de/gpw/2021/index.html>

- Own backup in a git repository

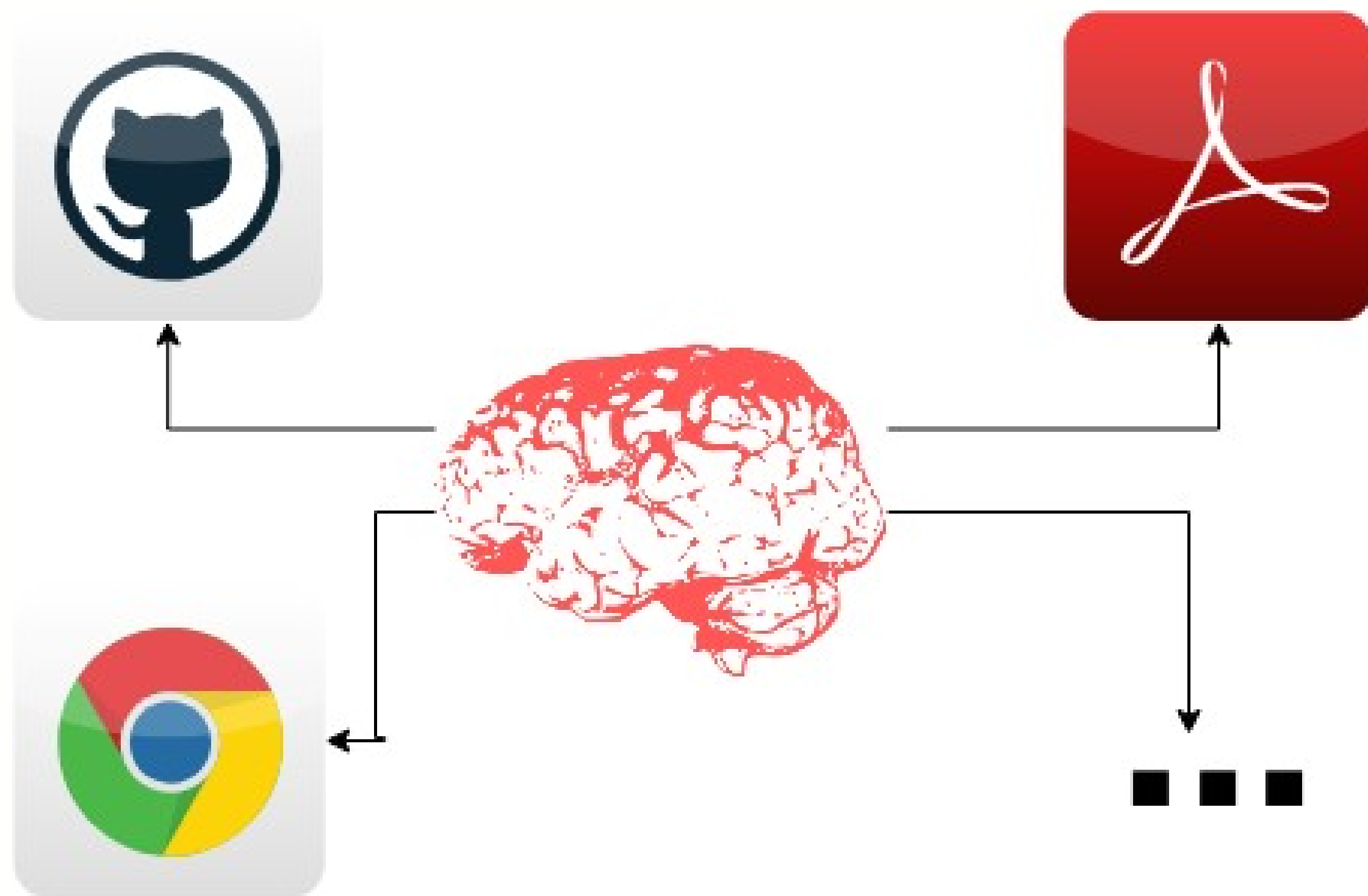


- Documentation as PDF files





- Documentation as HTML for customers
- Publish blog posts



Mojolicious and Nuclino

Mojolicious

- Nuclino is webbased
- ... Mojolicious has great tools to handle HTTP requests and responses
- ... we use Mojolicious in many projects → good knowledge

Mojolicious

- Nuclino is webbased
- ... Mojolicious has great tools to handle HTTP requests and responses
- ... we use Mojolicious in many projects → good knowledge

1) Unfortunately there's no (documented) API

1st question

- How can we get the contents?
 - Manually?
 - With a Perl script?

☰ Nuclino 🔍 CPANModules

List Board Graph Recent ⋮

CPANMODULES

OPM::Validate

Copy Test

Active

Archive

Trash

Workspace settings

Export

EXPORT WORKSPACE

<https://files.nuclino.com/export/brains/ace1472e-617a-4bd1-8ae4-05673374bb82.zip?format=md>

1st question

- How can we get the contents?
 - Manually?
 - With a Perl script?

1)Download zip archive for each „brain“

Get contents with Mojolicious

```
use Mojo::File qw(path);
use Mojo::JSON qw(decode_json);
use Mojo::UserAgent;
use Mojo::UserAgent::CookieJar;

my $ua = Mojo::UserAgent->new(
    cookie_jar      => Mojo::UserAgent::CookieJar->new,
    max_connections => 200,
);

my $config = decode_json path(__FILE__)->sibling('nuclino.json')->slurp;

my $header = {
    # header wie in den Developertools zu sehen
};
```


Get contents with Mojolicious

```
for my $brain ( @brains ) {  
    my $url = sprintf 'https://files.nuclino.com/export/brains/%s.zip?format=md', $brain;  
  
    my $tx_backup = $ua->get(  
        $url => $header  
    );  
  
    my $dir = path('/home/data/backups')->child( $brain . '.zip' );  
  
    $tx_backup->res->save_to( $zip_file );  
}
```

Get contents with Mojolicious

```
for my $brain ( @brains ) {  
    my $url = sprintf 'https://files.nuclino.com/export/brains/%s.zip?format=md', $brain;  
  
    my $tx_backup = $ua->get(  
        $url => $header  
    );  
  
    my $dir = path('/home/data/backups')->child( $brain . '.zip' );  
  
    $tx_backup->res->save_to( $zip_file );  
}
```

Get contents with Mojolicious

```
for my $brain ( @brains ) {  
    my $url = sprintf 'https://files.nuclino.com/export/brains/%s.zip?format=md', $brain;  
  
    my $tx_backup = $ua->get(  
        $url => $header  
    );  
  
    my $dir = path('/home/data/backups')->child( $brain . '.zip' );  
  
    $tx_backup->res->save_to( $zip_file );  
}
```

Get contents with Mojolicious

```
for my $brain ( @brains ) {  
    my $url = sprintf 'https://files.nuclino.com/export/brains/%s.zip?format=md', $brain;  
  
    my $tx_backup = $ua->get(  
        $url => $header  
    );  
  
    my $dir = path('/home/data/backups')->child( $brain . '.zip' );  
  
    $tx_backup->res->save_to( $zip_file );  
}
```

2nd question

- How can we get the brain id?
 - What requests are done in the browser?
 - With a Perl script?



Filter URLs

Status	Method	Domain	File
204	OPTIONS	api.nuclino.com	auth
200	POST	api.nuclino.com	auth
204	OPTIONS	api.nuclino.com	initial-state
200	GET	api.nuclino.com	initial-state
101	GET	api.nuclino.com	syncing
204	OPTIONS	api.nuclino.com	refresh-session
200	POST	api.nuclino.com	refresh-session
200	GET	www.google-analytics.com	collect?v=1&_v=j88&a=247351752&t=pagevi



Filter URLs

Status	Method	Domain	File
204	OPTIONS	api.nuclino.com	auth
200	POST	api.nuclino.com	auth
204	OPTIONS	api.nuclino.com	initial-state
200	GET	api.nuclino.com	initial-state
101	GET	api.nuclino.com	syncing
204	OPTIONS	api.nuclino.com	refresh-session
200	POST	api.nuclino.com	refresh-session
200	GET	www.google-analytics.com	collect?v=1&_v=j88&a=247351752&t=pagevi



Filter URLs

Status	Method	Domain	File
204	OPTIONS	api.nuclino.com	auth
200	POST	api.nuclino.com	auth
204	OPTIONS	api.nuclino.com	initial-state
200	GET	api.nuclino.com	initial-state
101	GET	api.nuclino.com	syncing
204	OPTIONS	api.nuclino.com	refresh-session
200	POST	api.nuclino.com	refresh-session
200	GET	www.google-analytics.com	collect?v=1&_v=j88&a=247351752&t=pagevi

2nd question

- How can we get the brain id?
 - What requests are done in the browser?
 - With a Perl script?

1) Login

2) Get initial state
→ Get brain IDs

Get brain IDs with Mojolicious

```
my $base_url = 'https://api.nuclino.com/api/users/';
my $tx_init  = $ua->get('https://app.nuclino.com/login');
my $tx_login = $ua->post(
    $base_url . 'auth' => $header => json => $config,
);

my $tx_initial_state = $ua->get(
    $base_url . 'me/initial-state' => $header
);

my @brains = @{
    $tx_initial_state->res->json('/response/teams/0/data/brains') || []
};
```

Get brain IDs with Mojolicious

```
my $base_url = 'https://api.nuclino.com/api/users/';
my $tx_init  = $ua->get('https://app.nuclino.com/login');
my $tx_login = $ua->post(
    $base_url . 'auth' => $header => json => $config,
);

my $tx_initial_state = $ua->get(
    $base_url . 'me/initial-state' => $header
);

my @brains = @{
    $tx_initial_state->res->json('/response/teams/0/data/brains') || []
};
```

Get brain IDs with Mojolicious

```
my $base_url = 'https://api.nuclino.com/api/users/';
my $tx_init  = $ua->get('https://app.nuclino.com/login');
my $tx_login = $ua->post(
    $base_url . 'auth' => $header => json => $config,
);

my $tx_initial_state = $ua->get(
    $base_url . 'me/initial-state' => $header
);

my @brains = @{
    $tx_initial_state->res->json('/response/teams/0/data/brains') || []
};
```

Get brain IDs with Mojolicious

```
my $base_url = 'https://api.nuclino.com/api/users/';
my $tx_init = $ua->get('https://app.nuclino.com/login');
my $tx_login = $ua->post(
    $base_url . 'auth' => $header => json => $config,
);

my $tx_initial_state = $ua->get(
    $base_url . 'me/initial-state' => $header
);

my @brains = @{
    $tx_initial_state->res->json('/response/teams/0/data/brains') || []
};
```

```
{
  "response": {
    "userId": "<user-uuid>",
    "teams": [
      {
        "id": "<team-uuid>",
        "data": {
          "name": "<team-name>",
          "brains": [
            "734eaf2c-b066-11ea-81bd-a36778484b6c",
            "820c6310-b066-11ea-9648-bbd09f732919",
            ...
          ]
        }
      },
      ...
    ]
  },
  ...
}
```

Get contents with Mojolicious

```
for my $brain ( @brains ) {  
    my $url = sprintf 'https://files.nuclino.com/export/brains/%s.zip?format=md', $brain;  
  
    my $tx_backup = $ua->get(  
        $url => $header  
    );  
  
    my $dir = path('/home/data/backups')->child( $brain . '.zip' );  
  
    $tx_backup->res->save_to( $zip_file );  
}
```


Problem!

to replace links, we need a mapping for these cases:

- <https://app.nuclino.com/GPW2021/Talks/Liste-der-Talks-2a9d5bf7-bd59-42ab-8ae6-e46331e969ff>
- <https://app.nuclino.com/t/b/29ad1dec-8bed-4480-98ec-58f19ed993c9>
- <https://app.nuclino.com/GPW2021/Maßnahmen>
- [Nuclino deadaffe.md]

3rd question

- How can we get a mapping id ↔ title?
 - With a Perl script?

Goal: Mapping id ↔ title

```
{
  "brains" : {
    "0184e034-3156-11eb-bf6d-2bbce7395d9b" : "Kunden",
    "0f0b2e2a-3156-11eb-a397-97d9e025800c" : "FeatureAddons",
    "14c45b66-3156-11eb-bf04-835bcd7f7bdc7" : "PerlAcademy",
    ...
  },
  "documents" : {
    "19c8e5e6-3156-11eb-ba6d-8beba50b7276" : {
      "file": "index.html",
      "title": "index.html"
    },
    "27fc869a-3156-11eb-98f5-8f9deed3e298" : {
      "file": "GraphQL vs. REST 0051eb4b.md",
      "title": "GraphQL vs. REST"
    },
    ...
  },
  ...
}
```



Filter URLs

Status	Method	Domain	File
204	OPTIONS	api.nuclino.com	auth
200	POST	api.nuclino.com	auth
204	OPTIONS	api.nuclino.com	initial-state
200	GET	api.nuclino.com	initial-state
101	GET	api.nuclino.com	syncing
204	OPTIONS	api.nuclino.com	refresh-session
200	POST	api.nuclino.com	refresh-session
200	GET	www.google-analytics.com	collect?v=1&_v=j88&a=247351752&t=pagevi

The screenshot shows the Chrome DevTools Network tab. The top toolbar includes icons for Inspector, Console, Debugger, Style Editor, Performance, Memory, Network (active), Storage, Accessibility, and Application. Below the toolbar is a filter bar with 'Filter URLs' and a list of filter types: All, HTML, CSS, JS, XHR, Fonts, Images, Media, WS, Other. A 'Disable Cache' checkbox is checked, and 'No Throttling' is selected. The main table lists network requests with columns: Status, Method, Domain, File, Initiator, Type, Transferred, and Size. The selected request is a GET request to 'api.nuclino.com' for the file 'syncing', with a status of 101 and a size of 233 B. The right pane shows the 'Response' tab for this request, displaying a JSON object. The JSON structure is as follows:

```
ns: "sd"
data: {
  snapshotMap: {
    "076d01b2-faa7-4b33-816c-a2d56fe72888": {
      id: "076d01b2-faa7-4b33-816c-a2d56fe72888"
      v: 16
      type: "https://nuclino.com/ot-types/json01"
      data: {
        kind: "LEAF"
        title: "Modulverwendung messen"
        fields: {}
        brainId: "3e5ff097-a236-4966-a1ba-5352deaa1636"
      }
      sharing: {
        mode: null
        token: null
      }
    }
  }
}
```

Create mapping

```
sub _create_mapping {  
    my ( $ua, $info, $brains, $home_dir ) = @_;  
  
    my $mapping = {};  
  
    my $promise = Mojo::Promise->new;  
  
    $ua->websocket(  
        'wss://api.nuclino.com/syncing' => {  
            Via => '1.1 vegur',  
            Origin => 'https://app.nuclino.com',  
        } => ['v1.proto'] => sub {  
  
            # ... hier der Code fuer den Informationsaustausch  
  
        });  
}
```

Create mapping

```
sub _create_mapping {  
    my ( $ua, $info, $brains, $home_dir ) = @_;  
  
    my $mapping = {};  
  
    my $promise = Mojo::Promise->new;  
  
    $ua->websocket(  
        'wss://api.nuclino.com/syncing' => {  
            Via => '1.1 vegur',  
            Origin => 'https://app.nuclino.com',  
        } => ['v1.proto'] => sub {  
  
            # ... hier der Code fuer den Informationsaustausch  
  
        });  
}
```

Create mapping

```
my ($ua, $tx) = @_;  
  
return if !$tx->isa('Mojo::Transaction::WebSocket');  
  
$tx->on( message => sub ($tx, $msg) {  
    _handle_message( $tx, $msg, $mapping );  
});  
  
$tx->on( finish => sub ($tx) {  
    # generate JSON file for mapping  
    $home_dir->child('backups', 'mapping.json')->spurt(  
        encode_json $mapping  
    );  
  
    $promise->resolve(1);  
});
```

Create mapping

```
my ($ua, $tx) = @_;  
  
return if !$tx->isa('Mojo::Transaction::WebSocket');  
  
$tx->on( message => sub ($tx, $msg) {  
    _handle_message( $tx, $msg, $mapping );  
});  
  
$tx->on( finish => sub ($tx) {  
    # generate JSON file for mapping  
    $home_dir->child('backups', 'mapping.json')->spurt(  
        encode_json $mapping  
    );  
  
    $promise->resolve(1);  
});
```


Create mapping

```
my ($ua, $tx) = @_;  
  
return if !$tx->isa('Mojo::Transaction::WebSocket');  
  
$tx->on( message => sub ($tx, $msg) {  
    _handle_message( $tx, $msg, $mapping );  
});  
  
$tx->on( finish => sub ($tx) {  
    # generate JSON file for mapping  
    $home_dir->child('backups', 'mapping.json')->spurt(  
        encode_json $mapping  
    );  
  
    $promise->resolve(1);  
});
```


Create mapping

```
$tx->send( encode_json +{  
    ns => 'sd',  
    data => {  
        a => 's',  
        c => 'ot_config',  
        d => 'a16b46be-31a7-11eb-ad70-e338b15b7ae1',  
    }  
});
```

```
# hier noch die Requests fuer die Kommandos  
# ot_user, ot_user_private, ot_team
```

Create mapping

```
Mojo::IOLoop->timer( 3 => sub {  
    for my $brain ( @{$brains} ) {  
        $tx->send( encode_json +{  
            ns => 'sd',  
            data => {  
                a => 's',  
                c => 'ot_brain',  
                d => $brain,  
            }  
        });  
    }  
});
```

Create mapping

```
Mojo::IOLoop->timer( 3 => sub {  
    for my $brain ( @{$brains} ) {  
        $tx->send( encode_json +{  
            ns => 'sd',  
            data => {  
                a => 's',  
                c => 'ot_brain',  
                d => $brain,  
            }  
        });  
    }  
});
```

```
{  
  "ns": "sd",  
  "data": {  
    "data": {  
      "v": 2,  
      "data": {  
        "kind": "PUBLIC",  
        "name": "Werkzeuge",  
        "teamId": "1f2966e2-321d-11eb-8f00-5b043dffbd53",  
        "members": [],  
        "createdAt": "2019-07-17T15:40:28.897Z",  
        "creatorId": "2a9e9ad8-321d-11eb-a81f-631faf46d6c4",  
        "mainCellId": "320c3df2-321d-11eb-9604-fbfbbed008d34",  
        "defaultView": "TREE",  
        "trashCellId": "377bf386-321d-11eb-b969-3fea1f109107",  
        "archiveCellId": "3de4e624-321d-11eb-a864-67366681e78f",  
        "formerMembers": [],  
        "pinnedCellIds": [],  
        "defaultMemberRole": "MEMBER"  
      },  
      "type": "https://nuclino.com/ot-types/json01"  
    },  
    "a": "s",  
    "c": "ot_brain",  
    "d": "7091aff1-1513-4e68-ae9f-d6f8936fcf14"  
  }  
}
```

Create mapping

```
$tx->send( encode_json +{  
    ns => 'sd',  
    data => {  
        a => 's',  
        c => 'ot_cell',  
        d => $cell,  
    }  
});
```

Create mapping

```
$tx->send( encode_json +{  
    ns => 'sd',  
    data => {  
        a => 's',  
        c => 'ot_cell',  
        d => $cell,  
    }  
});
```

```
{  
  "ns": "sd",  
  "data": {  
    "data": {  
      "v": 18,  
      "data": {  
        "kind": "MAIN",  
        "title": "Main",  
        "brainId": "a29de3ff-18a5-4d51-ab9b-a6c3b5d82b6c",  
        "sharing": {},  
        "childIds": [  
          "bb10e948-ba91-4097-83a1-60c9ea6ec17b",  
          "4d44737d-e113-4786-bb98-06d8755d2bb7",  
          "..."  
        ],  
        "createdAt": "2019-07-17T14:47:35.743Z",  
        "creatorId": "34e53829-cafb-4859-9e7b-0aae34295d04",  
        "memberIds": [],  
        "updatedAt": "2019-07-17T14:47:35.743Z",  
        "activities": [],  
        "contentMeta": {}  
      },  
      "type": "https://nuclino.com/ot-types/json01"  
    },  
    "a": "s",  
    "c": "ot_cell",  
    "d": "d56942a8-aa6e-4197-93fd-1f88967dedc6"  
  }  
}
```

Thank you!

Questions?

Slides available at <https://gitlab.com/perl-academy/talks/2021/gpw/>