Growing a Tech Team: Autonomy vs Anarchy

Sofía LESCANO CARROLL

@SofLesc

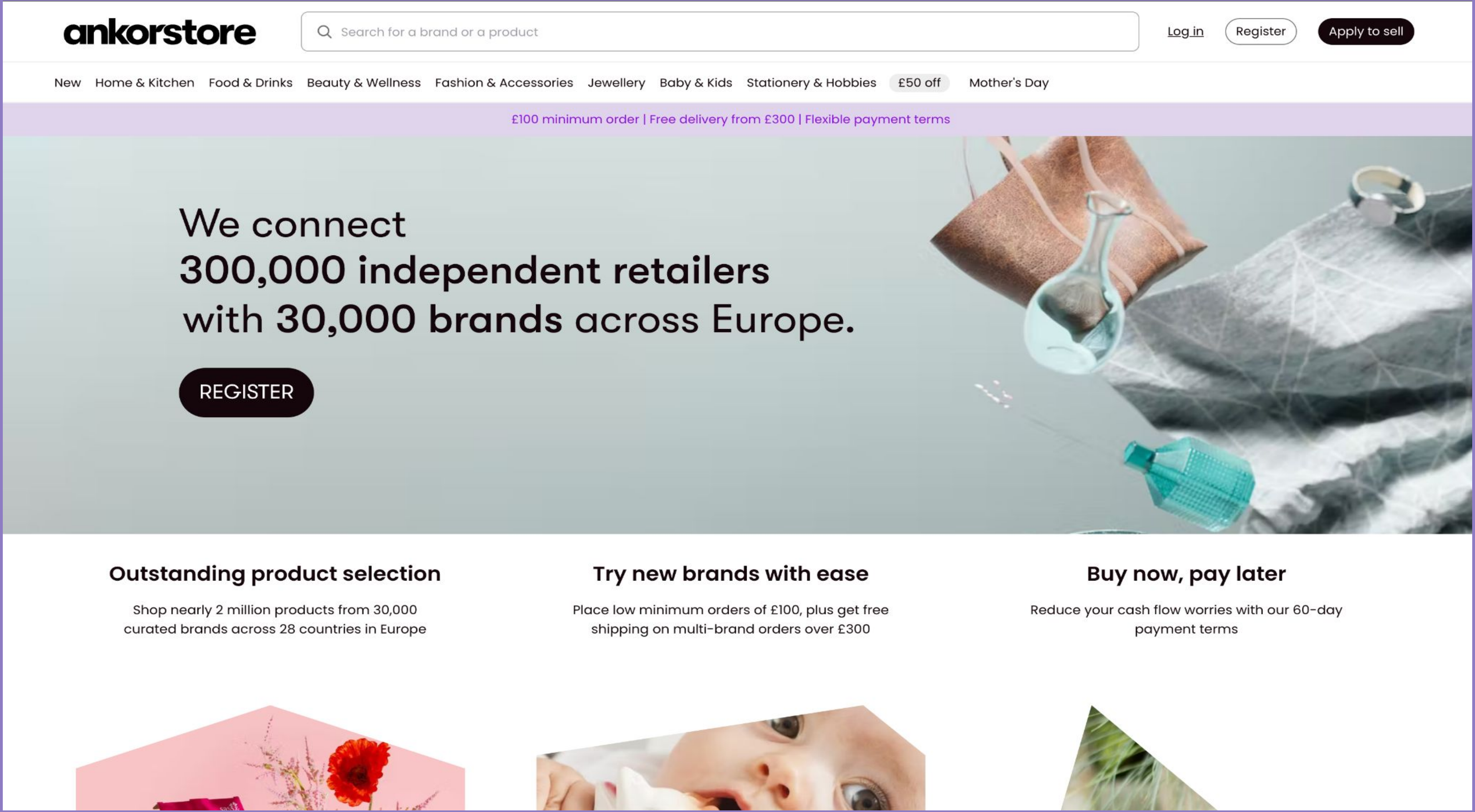
Sofia Lescano Carroll

Senior Software Developer

Madrid

ankorstore

We rewild retail
2022



ankorstore

Search for a brand or a product

Log in | Register | Apply to sell

New | Home & Kitchen | Food & Drinks | Beauty & Wellness | Fashion & Accessories | Jewellery | Baby & Kids | Stationery & Hobbies | £50 off | Mother's Day

£100 minimum order | Free delivery from £300 | Flexible payment terms

We connect
300,000 independent retailers
with **30,000 brands** across Europe.

REGISTER

Outstanding product selection

Shop nearly 2 million products from 30,000 curated brands across 28 countries in Europe

Try new brands with ease

Place low minimum orders of £100, plus get free shipping on multi-brand orders over £300

Buy now, pay later

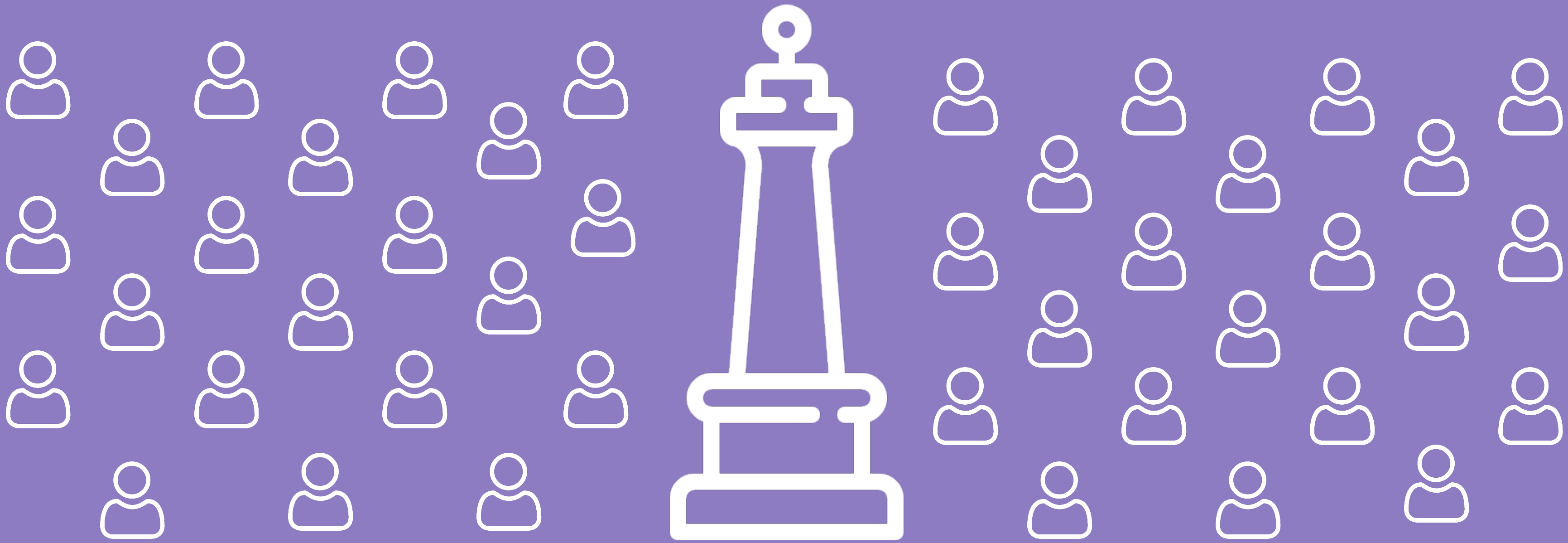
Reduce your cash flow worries with our 60-day payment terms

A growing team : new challenges

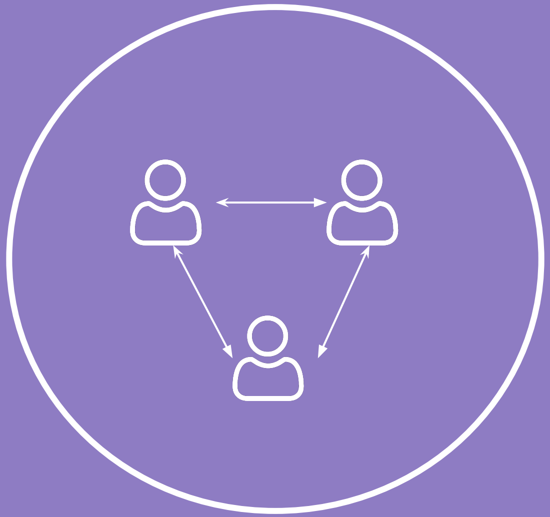
Product & tech nov'21

Product & tech nov'22

Our stack: a monolith

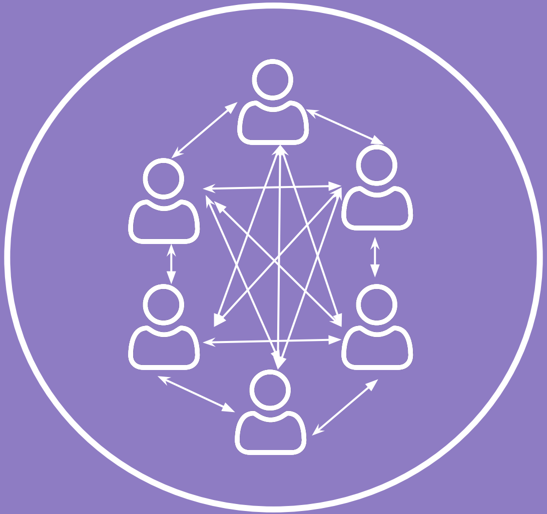


A growing team : new challenges



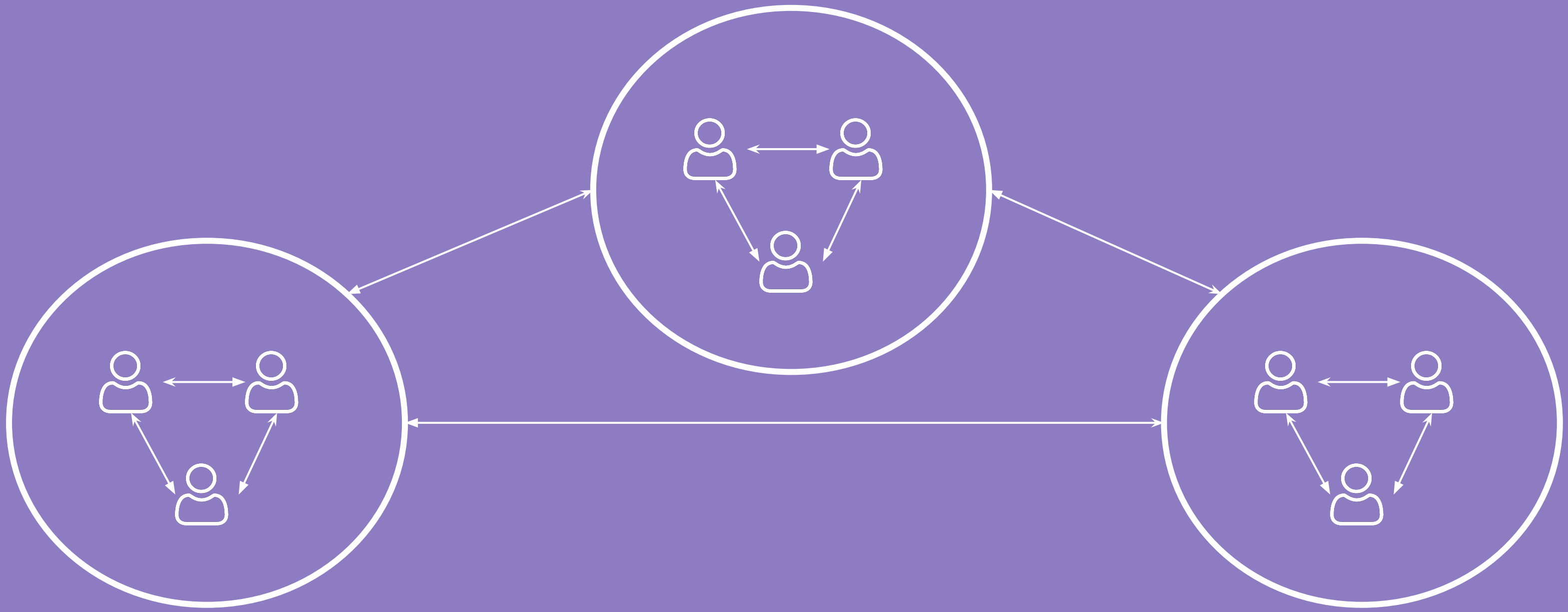
One team

A growing team : new challenges



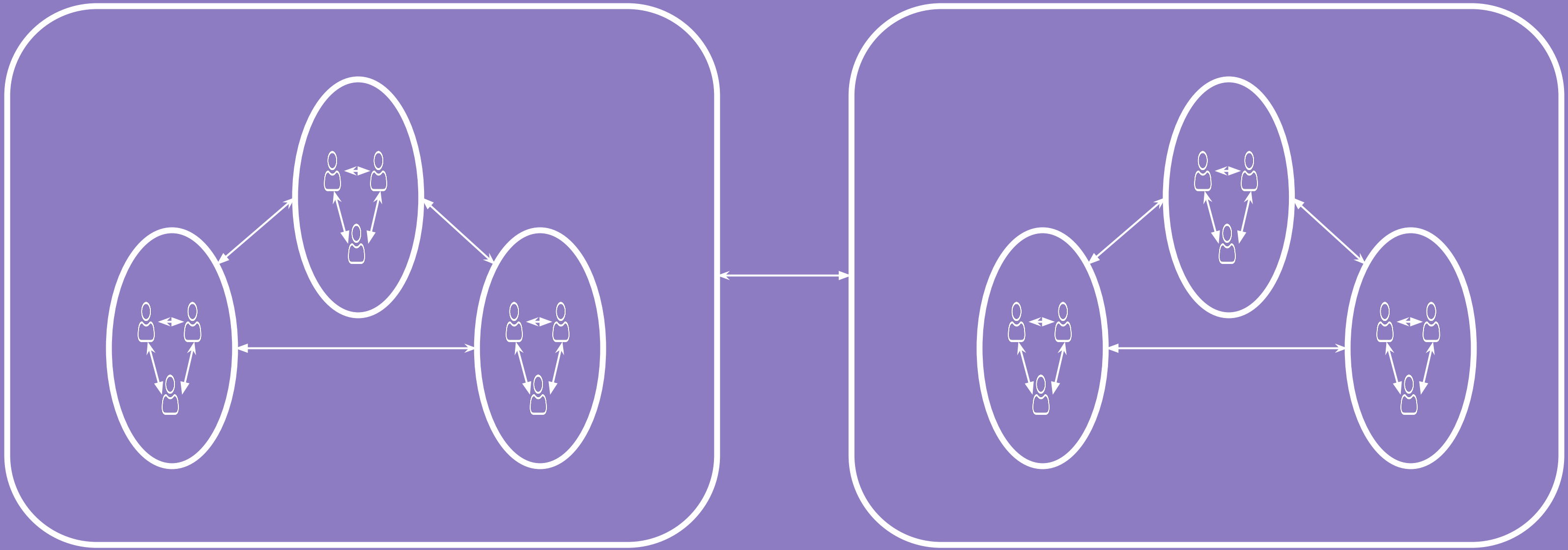
One bigger team

A growing team : new challenges



Multiple squads

A growing team : new challenges



Multiple tribes

As the team grows, the coordination problem grows. We need to coordinate initiatives and the tech stack across teams

A growing team : new challenges

Autonomy

VS

Anarchy

Working as a team

Do you work in a team or do you work alone?

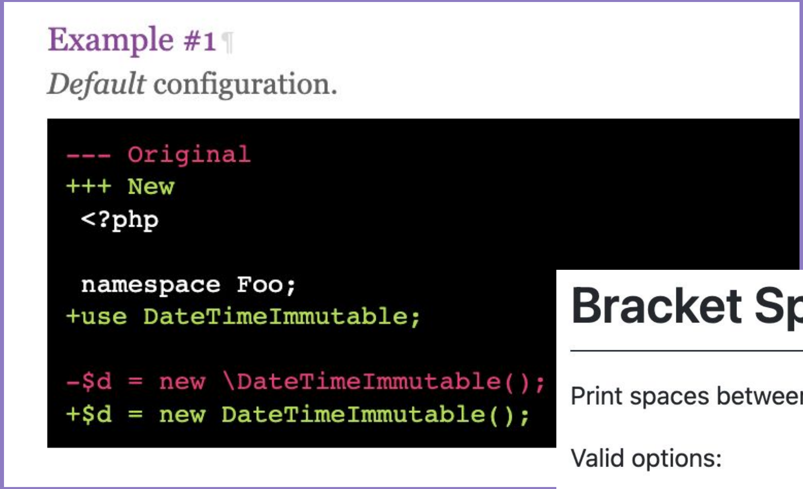
When we work alone we are in control of our choices

(and responsible for our bad choices)

When we work in a team, it **must** take precedence over our personal convictions

You know, I don't think there are good or bad technical choices. Me, if I had to sum up my life today with you, I would say that it is first of all choices.

@SofLesc



Example #1 ¶

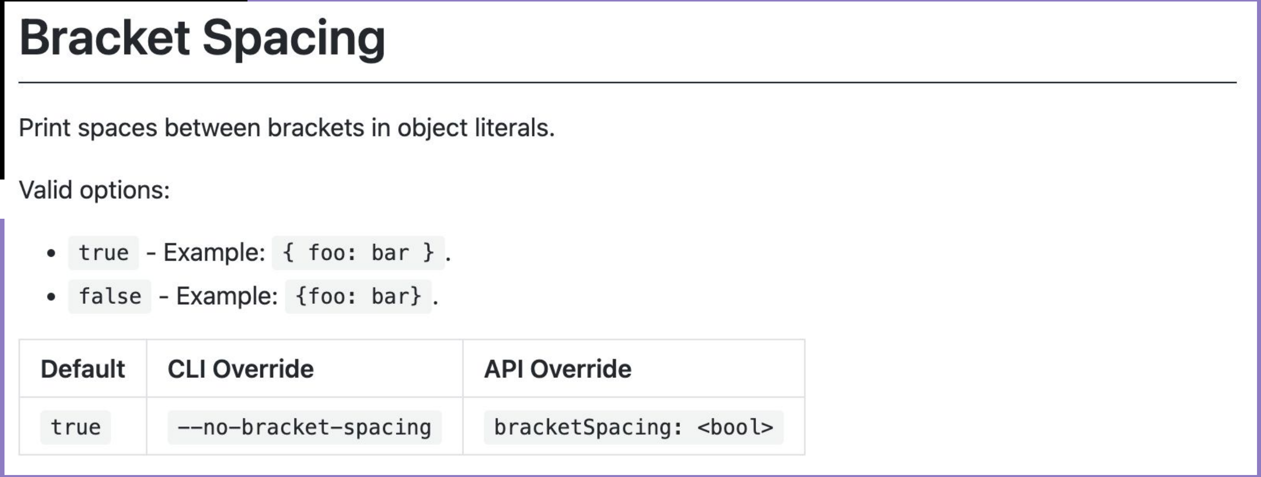
Default configuration.

```
--- Original
+++ New
 <?php

 namespace Foo;
+use DateTimeImmutable;

-$d = new \DateTimeImmutable();
+$d = new DateTimeImmutable();
```

PHP CS Fixer



Bracket Spacing

Print spaces between brackets in object literals.

Valid options:

- `true` - Example: `{ foo: bar }`.
- `false` - Example: `{foo: bar}`.

Default	CLI Override	API Override
`true`	`--no-bracket-spacing`	`bracketSpacing: <bool>`

Prettier

Code styling

Symfony

vs

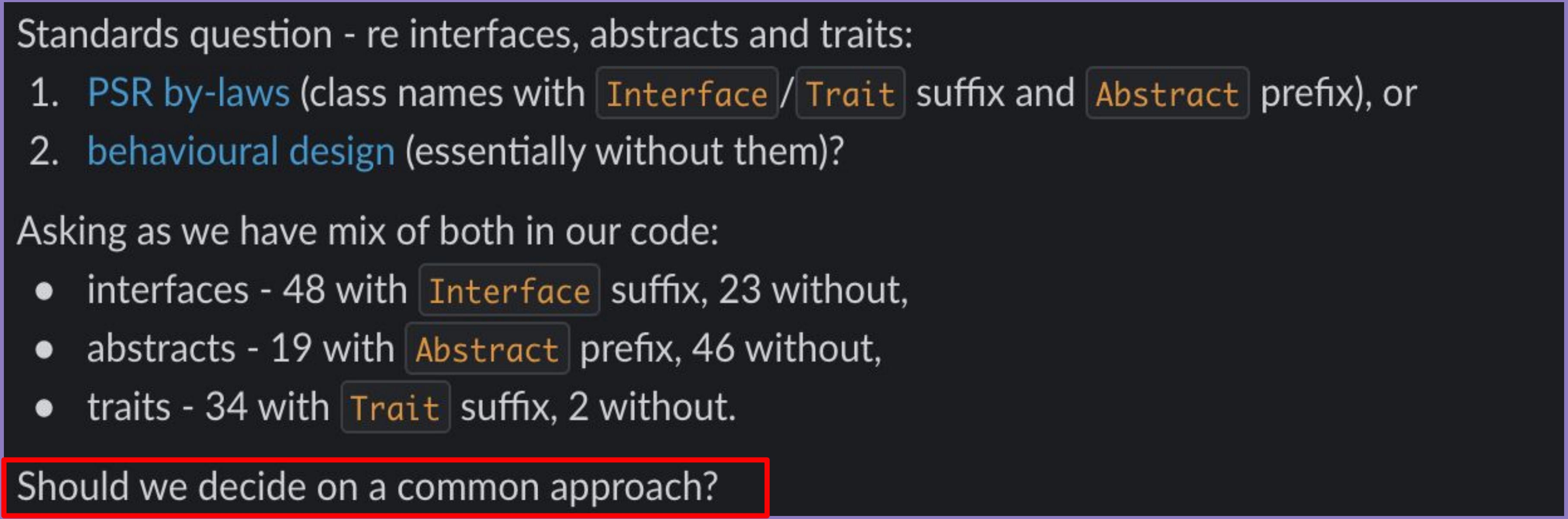
Laravel

Framework

Should we create an interface when there is only one implementation ?

Should we suffix interfaces with "interface" ?

Interfaces


Standards question - re interfaces, abstracts and traits:

1. PSR by-laws (class names with `Interface` / `Trait` suffix and `Abstract` prefix), or
2. behavioural design (essentially without them)?

Asking as we have mix of both in our code:

- interfaces - 48 with `Interface` suffix, 23 without,
- abstracts - 19 with `Abstract` prefix, 46 without,
- traits - 34 with `Trait` suffix, 2 without.

Should we decide on a common approach?

Interfaces

There is <u>no universal truth</u>, there are <u>choices</u> (standards, best practices, company guidelines, etc.). The choices are made for <u>the good of the collective</u> beyond individual sensitivities.

Our organisation

Our teams

Functional perimeter

Engineering manager

3-8 Software Engineers (BE & FE)

Product designer

Data engineer

Product manager

Our objective

$$\underbrace{\text{Sustainably deliver}}_{\text{Engineering}}\ \underbrace{\text{business value}}_{\text{Product}}$$

The Product Manager is largely responsible for “What to do”, and the rest of the team is responsible for “How to do it”.

Our guidelines

Engineering principles
&
Software architecture principles

Engineering principles

Safety

Transparency

Small long-lived squad

Autonomy

Collaboration

Self-organization

Iterative process

Engineering principles

Software architecture principles

Data privacy

Always improve

Environment parity

Testing

Localisation

Documentation

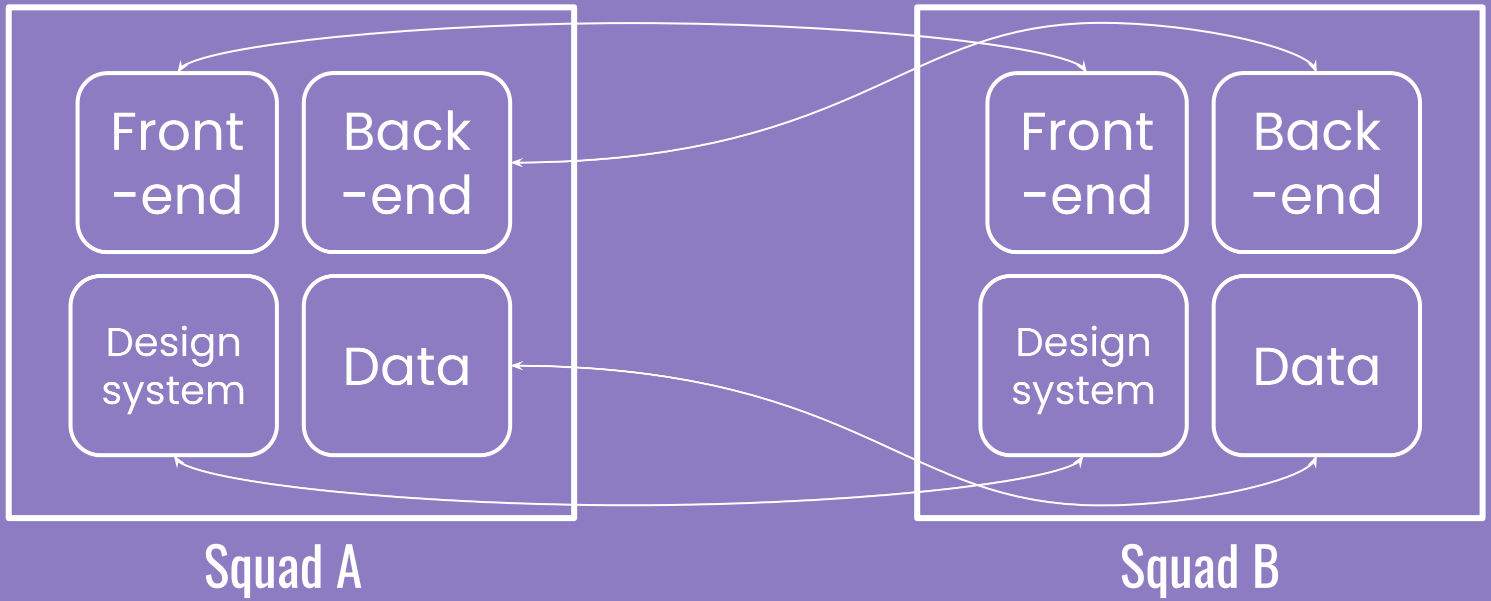
No deviant system behaviour

Disposability

Software architecture principles

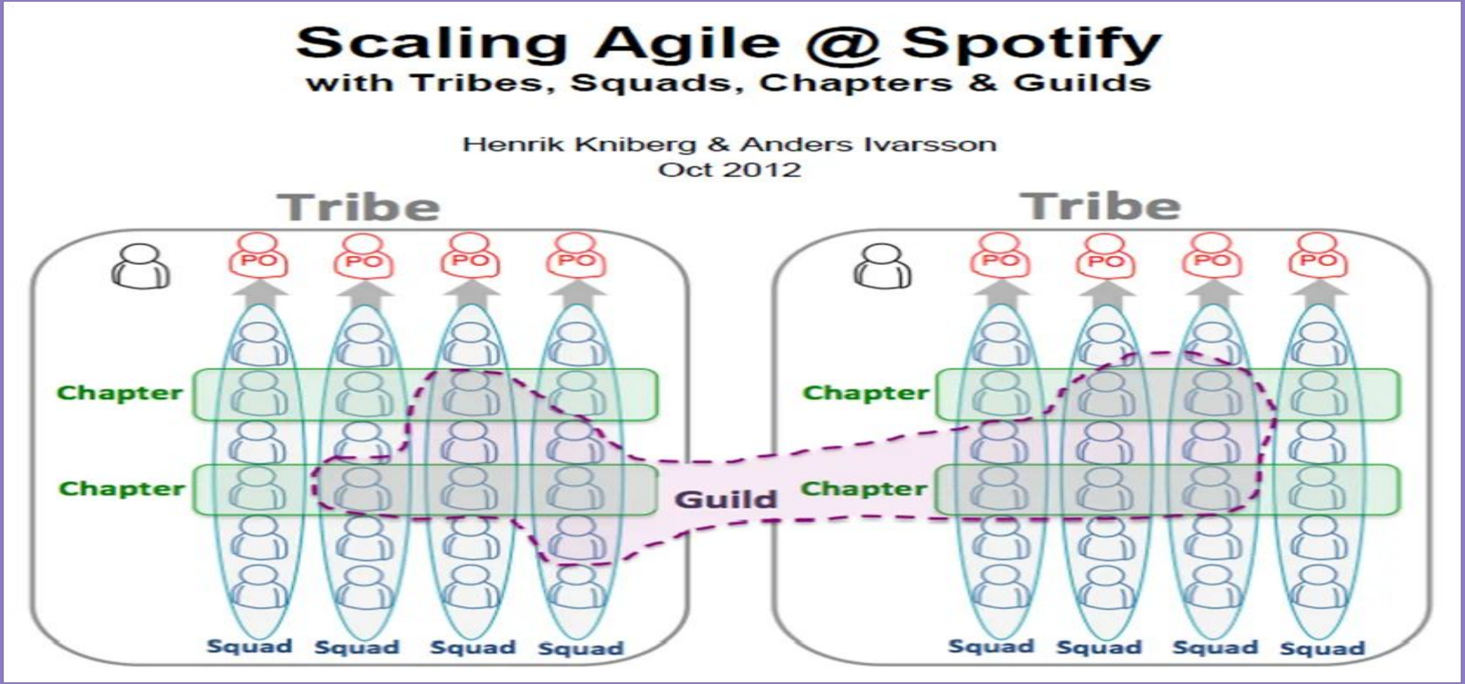
Request For Comments
& Communities of Practice

RFC & CoP



Coordination between layers

RFC & CoP



Coordination between layers

RFC & CoP

Alignment

Volunteer

Open to everyone

Cross-squads

RFC & CoP

If you do not participate, decisions will be made without you

Request for comments

Request for comments

Decide together

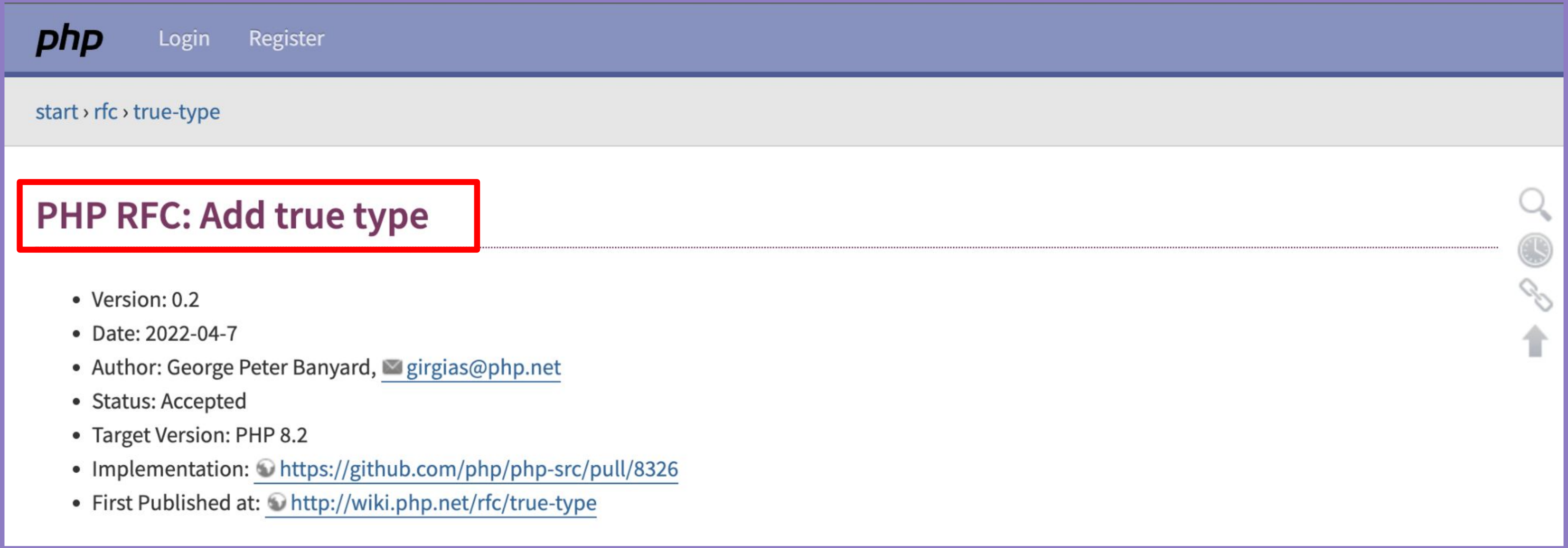
Simple and iterative

Widely used process

Participation is encouraged

Request for comments : structure

Request for comments



php Login Register

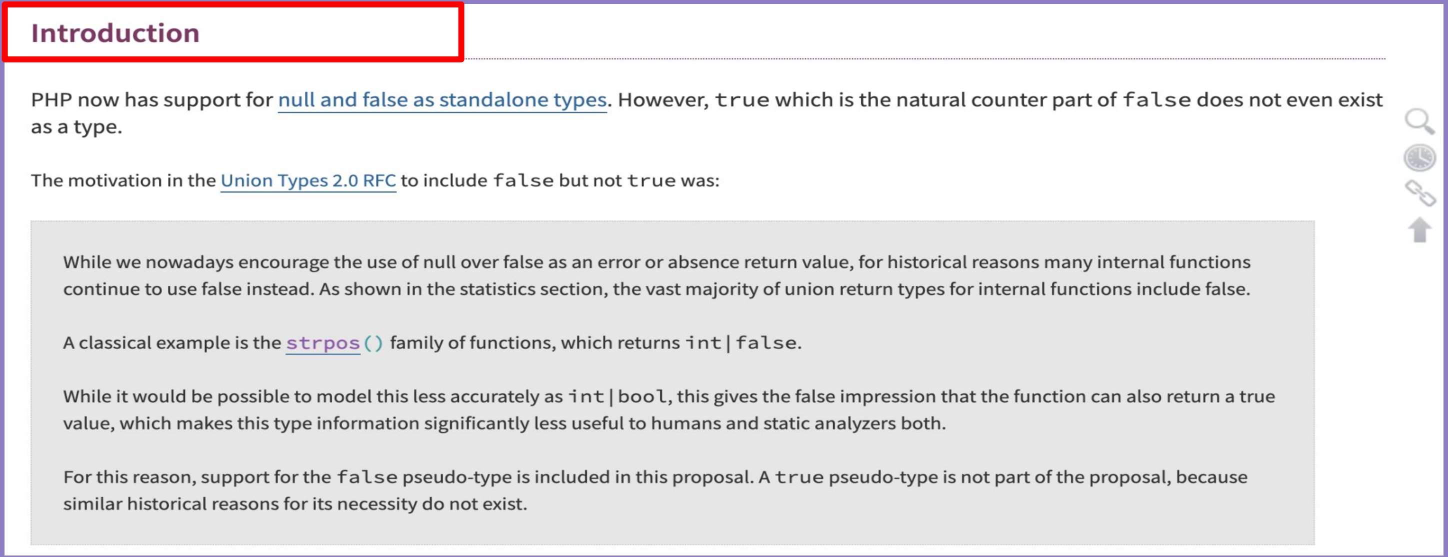
start › rfc › true-type

PHP RFC: Add true type

- Version: 0.2
- Date: 2022-04-7
- Author: George Peter Banyard, girgias@php.net
- Status: Accepted
- Target Version: PHP 8.2
- Implementation: https://github.com/php/php-src/pull/8326
- First Published at: http://wiki.php.net/rfc/true-type

- Title: short and clear
- Theme : Frontend, Backend, Coding Rules ...

Request for comments



Introduction

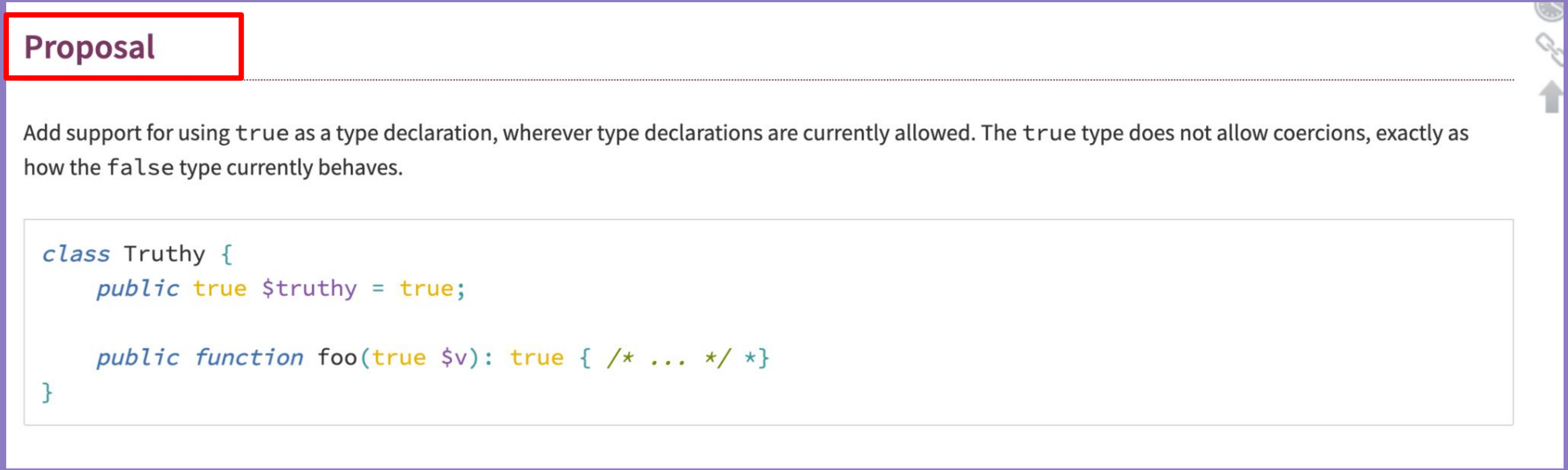
PHP now has support for null and false as standalone types. However, `true` which is the natural counter part of `false` does not even exist as a type.

The motivation in the Union Types 2.0 RFC to include `false` but not `true` was:

> While we nowadays encourage the use of null over false as an error or absence return value, for historical reasons many internal functions continue to use false instead. As shown in the statistics section, the vast majority of union return types for internal functions include false.
>
> A classical example is the `strpos()` family of functions, which returns `int|false`.
>
> While it would be possible to model this less accurately as `int|bool`, this gives the false impression that the function can also return a true value, which makes this type information significantly less useful to humans and static analyzers both.
>
> For this reason, support for the `false` pseudo-type is included in this proposal. A `true` pseudo-type is not part of the proposal, because similar historical reasons for its necessity do not exist.

- Summary:
  - What is the problem to solve?
  - What are the risks for the company if this problem is not solved?

Request for comments



Proposal

Add support for using `true` as a type declaration, wherever type declarations are currently allowed. The `true` type does not allow coercions, exactly as how the `false` type currently behaves.

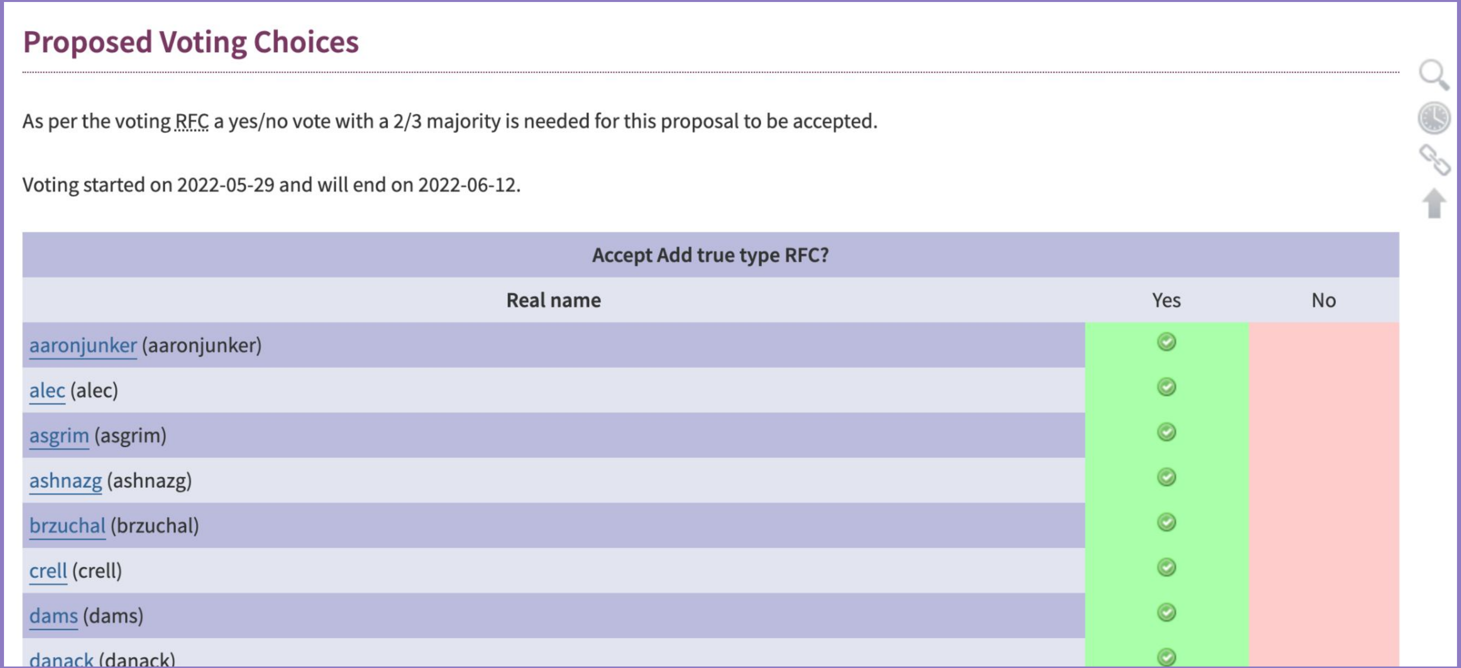
```
class Truthy {
    public true $truthy = true;

    public function foo(true $v): true { /* ... */ *}
}
```

- Proposal:
  - one or more
  - detailed : what ? how ? risks?

Try not to be **biased** when describing the solutions

Request for comments



Proposed Voting Choices

As per the voting RFC a yes/no vote with a 2/3 majority is needed for this proposal to be accepted.

Voting started on 2022-05-29 and will end on 2022-06-12.

Accept Add true type RFC?		
Real name	Yes	No
aaronjunker (aaronjunker)	✓	
alec (alec)	✓	
asgrim (asgrim)	✓	
ashnazg (ashnazg)	✓	
brzuchal (brzuchal)	✓	
crell (crell)	✓	
dams (dams)	✓	
danack (danack)	✓	

- Vote system

Request for comments : key actors

Key actors in an RFC

Owner(s)

Voters

Participants

Committee

Key actors in an RFC : owner(s)

Write RFC

Answer questions

Organise extra discussions

Key actors in an RFC: participants

Review

Help improve

Key actors in an RFC: voters

Read final RFC

Vote

Key actors in an RFC: comitee

Rotates

Ensure the RFC goes forward

Avoid biais

Assess impact

Key actors in an RFC: comitee

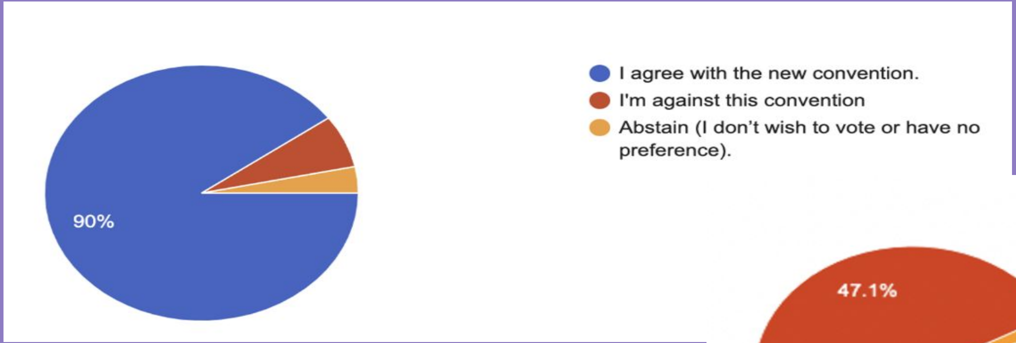
Select and RFC asignee

RFC examples

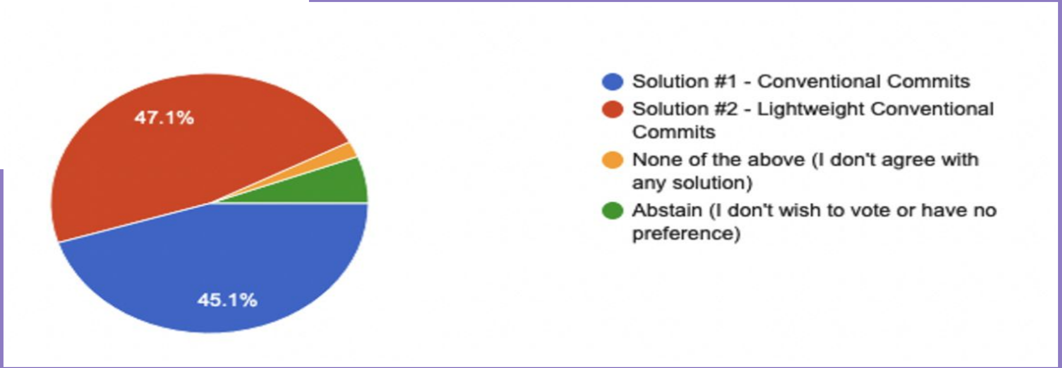
- Translation key convention
- Public Webhooks system
- Git commit messages convention
- External API architecture

RFC examples

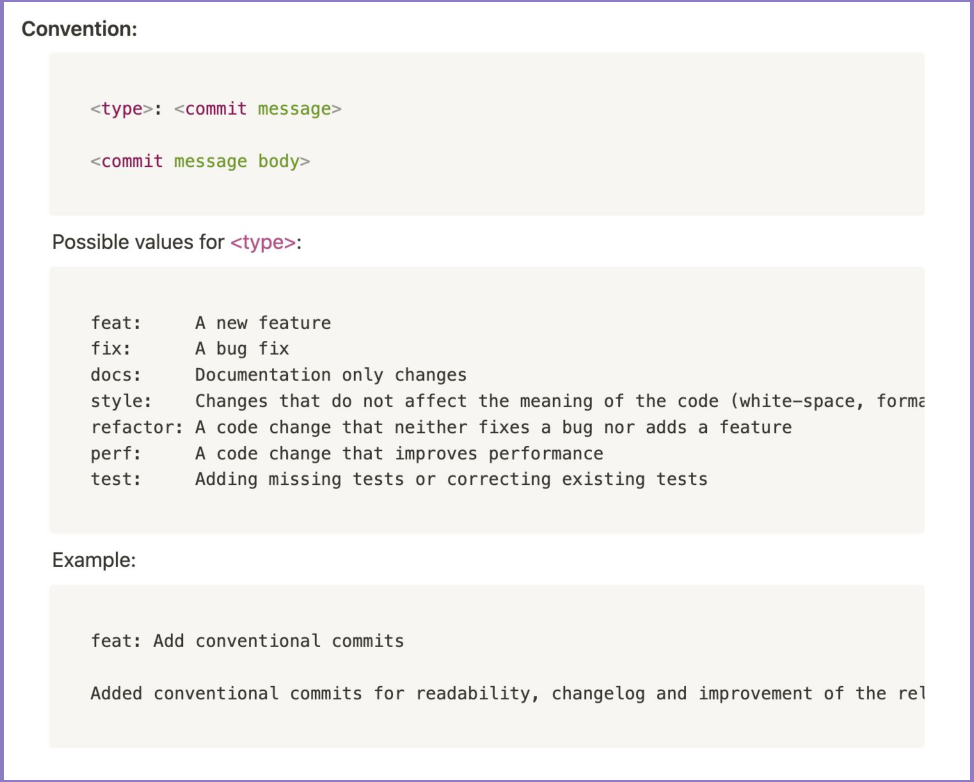
Translation key convention



Git commit messages convention



RFC examples: git commit messages



Convention:

```
<type>: <commit message>

<commit message body>
```

Possible values for <type>:

```
feat:     A new feature
fix:      A bug fix
docs:     Documentation only changes
style:    Changes that do not affect the meaning of the code (white-space, forma
refactor: A code change that neither fixes a bug nor adds a feature
perf:     A code change that improves performance
test:     Adding missing tests or correcting existing tests
```

Example:

```
feat: Add conventional commits

Added conventional commits for readability, changelog and improvement of the rel
```

Community of Practice

Community of Practice

Set of people that share a concern or a passion for something they do and learn together how to do it better

Community of Practice

Light-weight

Adaptive

Flexible

Consensus driven

Open to non-technical topics

Community of Practice

Backend

DDD

Testing

Front-End

DevOps

Design system

Engineering Manager

Community of Practice

Documentation

Recurring meetings

Contact point

Asynchronous discussions

Community of Practice

Tiny (1 ~ 10) : no structure

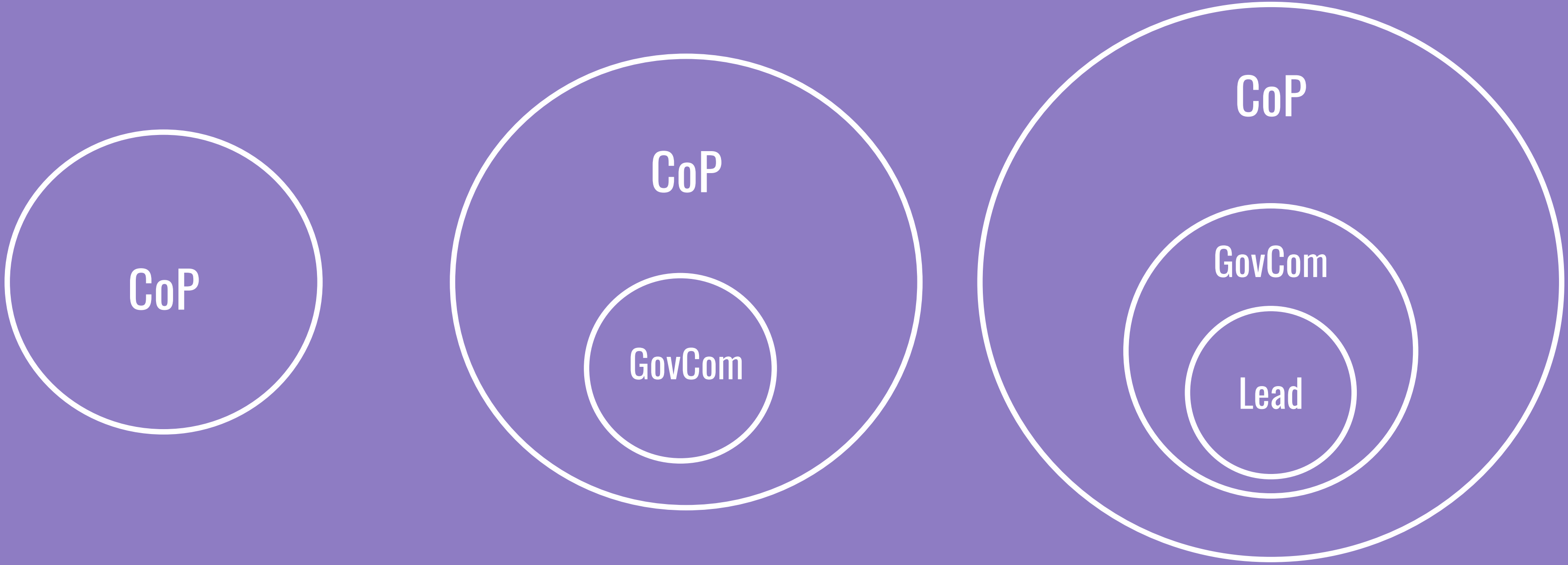
Small (10 ~ 25) : community of practice

Medium (25 ~ 50) : CoP + governance committee

Large (50 ~ 100) : CoP + GovCom + lead

Extra large (100+) : split into smaller groups

Community of Practice



Community of Practice: GovCom

Volunteers

~10% CoP

Rotate

Recurring meetings

Asynchronous discussions

Community of Practice: GovCom

Manage agenda

Decision making

Lead the CoP

Negotiate with hierarchy

Moderate discussion

My experience in the Backend CoP

Backend Chapter

Coffee with a staff engineer

Community of Practice: Backend

1H every two weeks

Online meeting

Everyone can bring a subject

Community of Practice: Backend

Share knowledge

Ask for feedback

Discuss new ideas

Community of Practice: Backend

Learn

Explore creativity

Take part in decisions

Challenges

Challenges

RFC adoption

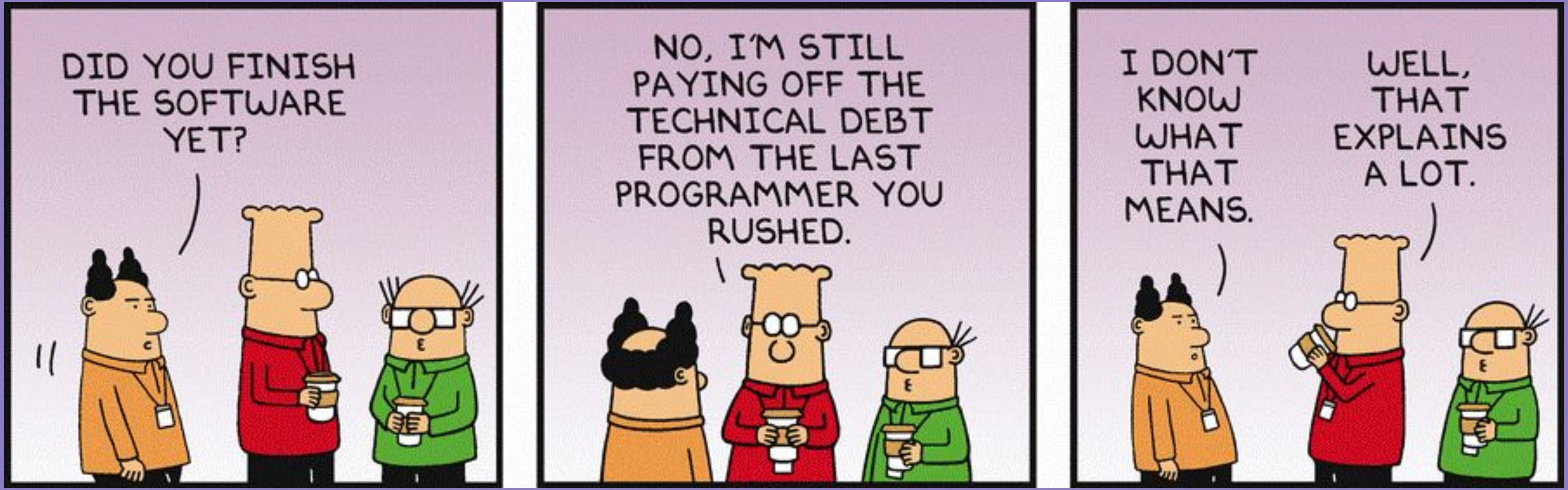
Features VS CoP/RFC

Find volunteers

“Looks nice, but this is not possible!”

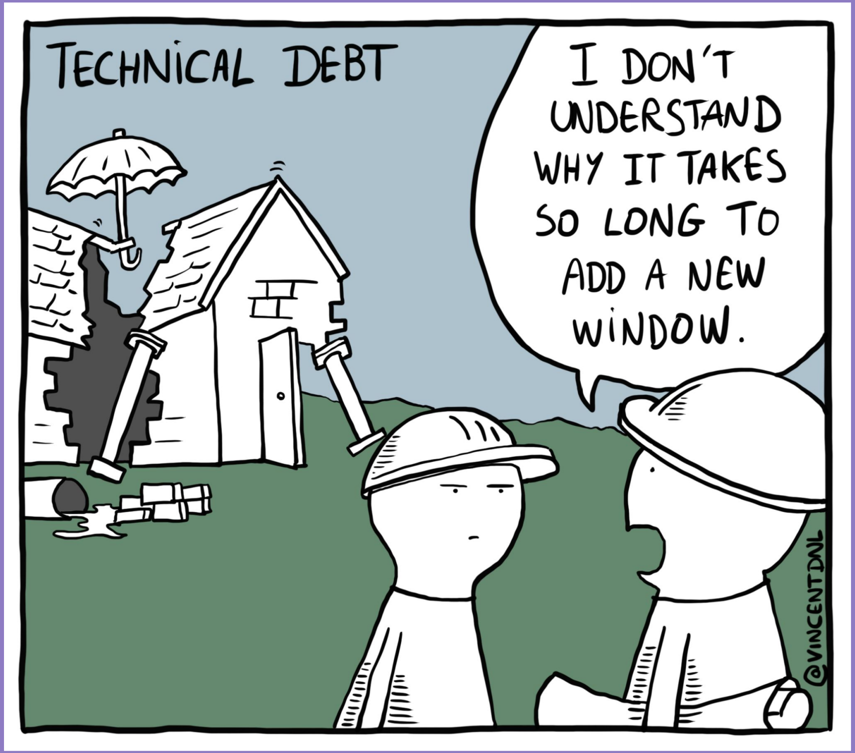
Risks

Risks


DID YOU FINISH THE SOFTWARE YET?
NO, I'M STILL PAYING OFF THE TECHNICAL DEBT FROM THE LAST PROGRAMMER YOU RUSHED.
I DON'T KNOW WHAT THAT MEANS.
WELL, THAT EXPLAINS A LOT.

@Dilbert_daily

Risks


TECHNICAL DEBT
I DON'T UNDERSTAND WHY IT TAKES SO LONG TO ADD A NEW WINDOW.
@VINCENTDNL

@VINCENTDNL

“Technical debt is the implied cost of additional rework caused by choosing an easy solution now instead of using a better approach that would take longer.”

Wikipedia

“Analogous with monetary debt, if technical debt is not repaid, it can accumulate *interest*, making it harder to implement changes.”

Wikipedia

“Technical debt is like when you want to cook dinner but first you have to do the dishes from the night before”

Olivier Mansour - Former Manager

@SofLesc

How to: tips

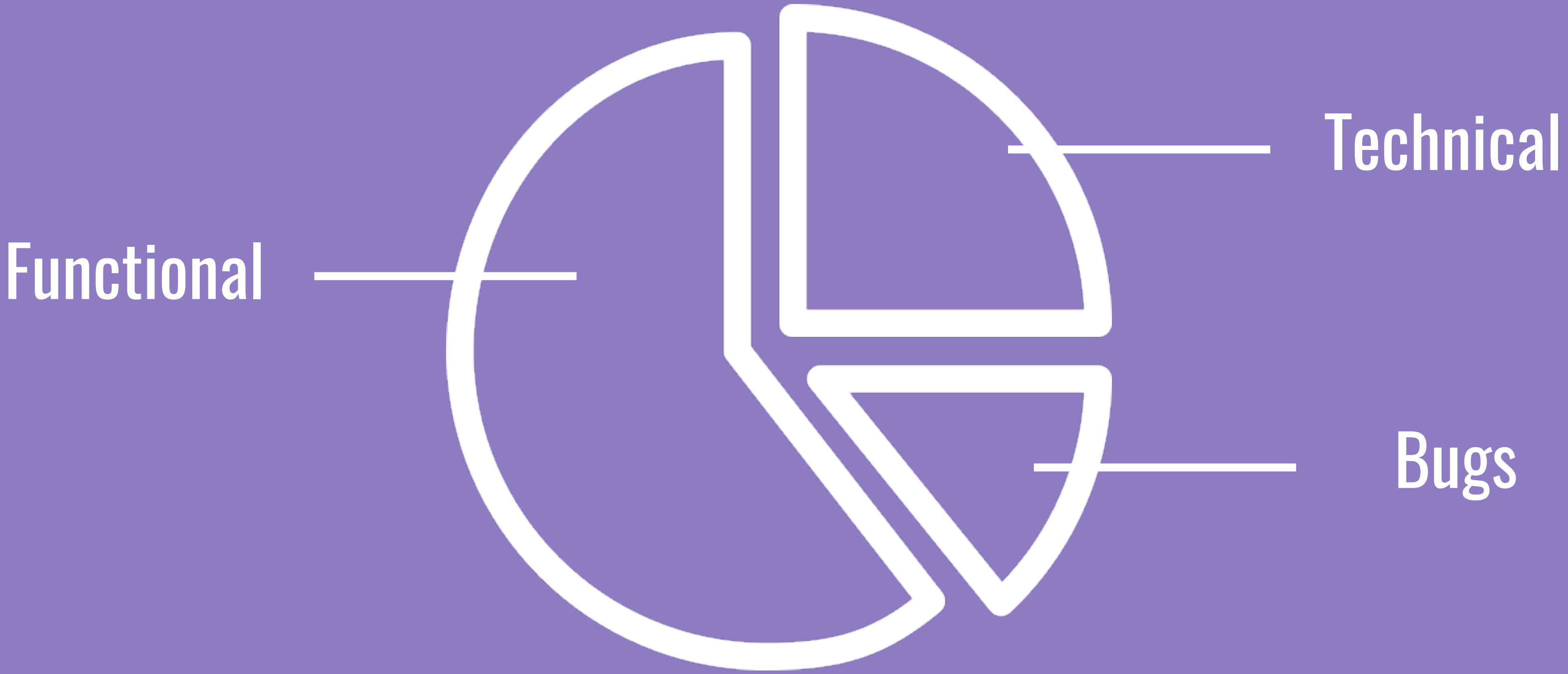
How to: win trust of stakeholders

Understand the stakes

Be transparent

Define engineering principles

How to: measure time spent



How to: see the benefits

Working confort

Platform stability

Add new features faster

Learn from each other

How to: make it your own way

Find your own recipe

Iterate

Uniconlabs

Eucalyp

Th studio

Freepik

Dimitri13

Noomtah

Bomsymbols

Phansan

Pixel perfect

Good ware

Thank you !

Zaenul Yahya

Turkkub

Maxim Basinski Premium

Vectorsmarket15

Fjstudio

Monkik

Icons made by the artists mentioned, from www.flaticon.com

Thank you !

Growing a Tech Team: Autonomy vs Anarchy

Sofía LESCANO CARROLL

@SofLesc

#fullRemote #PHP #Laravel #lifeInSpain #startup