Les Pratiques de Code : De la Documentation à la Détection

Défense de Thèse *par* Corentin Latappy *le* 19 Juin 2024

Université de Bordeaux - EDMI

Membres du Jury

M. FALLERI Jean-Rémy	Professeur des universités	Université de Bordeaux	Directeur de thèse
Mme ETIEN Anne	Professeure des universités	Université de Lille	Rapporteur
M. ACHER Mathieu	Professeur des universités	INSA Rennes	Rapporteur
M. ZEMMARI Akka	Professeur des universités	Université de Bordeaux	Examinateur
M. ZIMMERMANN Théo	Maître de Conférences	Télécom Paris	Examinateur

Membres invités

M. DEGUEULE Thomas	Chargé de recherche	CNRS	Co-encadrant
M. TEYTON Cédric	CTO	Packmind	Invité

université de BORDEAUX LaBRI

packmind

Les Pratiques de Code :
De la Documentation à la Détection

Les Pratiques de Code

```
4    function computeMean(data) {
5        let dataLength = data.length;
6
7        return data.reduce((acc, value) => acc + value, 0) / dataLength;
8    }
```

Les Pratiques de Code

```
4    function computeMean(data) {
5        const dataLength = data.length;
6
7        return data.reduce((acc, value) => acc + value, 0) / dataLength;
8    }
```

Les Pratiques de Code

```
4    function computeMean(data) {
5        let dataLength = data.length;
6
7        return data.reduce((acc, value) => acc + value, 0) / dataLength;
8    }
```

Convention

Les Pratiques de Code

```
4    function computeMean(data) {
5        let dataLength = data.length;
6
7        return data.reduce((acc, value) => acc + value, 0) / dataLength;
8    }
```

Convention

```
19   const userScript = window.prompt( message: 'Enter a script to execute:');
20
21   eval(userScript);
```

Sécurité

Les Pratiques de Code

```
4    function computeMean(data) {
5        let dataLength = data.length;
6
7        return data.reduce((acc, value) => acc + value, 0) / dataLength;
8    }
```

Convention

```
19   const userScript = window.prompt( message: 'Enter a script to execute:');
20
21   eval(userScript);
```

Sécurité

```
24   document.getElementById( elementId: 'header').style.color = 'red';
25   document.getElementById( elementId: 'header').style.backgroundColor = 'black';
26   document.getElementById( elementId: 'header').style.fontSize = '24px';
27   document.getElementById( elementId: 'header').style.fontWeight = 'bold';
28   document.getElementById( elementId: 'header').style.padding = '10px';
```

Performance

Les Pratiques de Code

```
4    function computeMean(data) {
5        let dataLength = data.length;
6
7        return data.reduce((acc, value) => acc + value, 0) / dataLength;
8    }
```

Convention

```
19   const userScript = window.prompt( message: 'Enter a script to execute:');
20
21   eval(userScript);
```

Sécurité

```
24   document.getElementById( elementId: 'header').style.color = 'red';
25   document.getElementById( elementId: 'header').style.backgroundColor = 'black';
26   document.getElementById( elementId: 'header').style.fontSize = '24px';
27   document.getElementById( elementId: 'header').style.fontWeight = 'bold';
28   document.getElementById( elementId: 'header').style.padding = '10px';
```

Performance

```
31   if(userScript.includes('alert')){alert('Not allowed');}else{eval(userScript);}
```

Lisibilité

Les Pratiques de Code

```
4   function computeMean(data) {
5       let dataLength = data.length;
6
7       return data.reduce((acc, value) => acc + value, 0) / dataLength;
8   }
```

Convention

```
19  const userScript = window.prompt( message: 'Enter a script to execute:');
20
21  eval(userScript);
```

Sécurité

```
24  document.getElementById( elementId: 'header').style.color = 'red';
25  document.getElementById( elementId: 'header').style.backgroundColor = 'black';
26  document.getElementById( elementId: 'header').style.fontSize = '24px';
27  document.getElementById( elementId: 'header').style.fontWeight = 'bold';
28  document.getElementById( elementId: 'header').style.padding = '10px';
```

Performance

```
31  if(userScript.includes('alert')){alert('Not allowed');}else{eval(userScript);}
```

Lisibilité

🚀 Qualité Logicielle

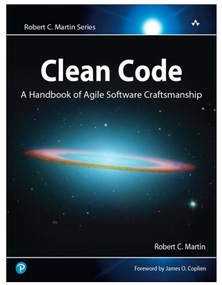
Les Pratiques de Code

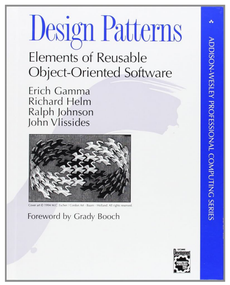
Origines

Les Pratiques de Code

Origines

📚 Livres

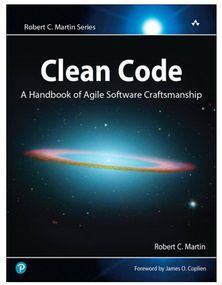


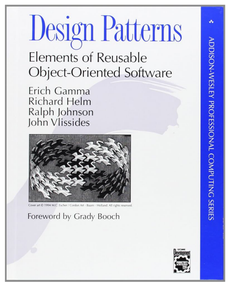


Les Pratiques de Code

Origines

📚 Livres





⌨️ Blogs

The Clean Code Blog

by Robert C. Martin (Uncle Bob)

Google Style Guides

Every major open-source project has its c
project. It is much easier to understand a

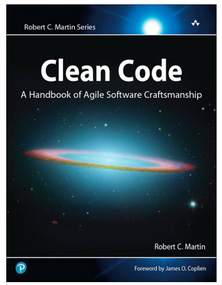
"Style" covers a lot of ground, from "use
project (google/styleguide) links to the st
you may be pointed to this page to see th

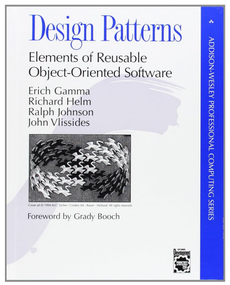
- AngularJS Style Guide
- Common Lisp Style Guide

Les Pratiques de Code

Origines

📚 Livres





⌨️ Blogs

The Clean Code Blog

by Robert C. Martin (Uncle Bob)

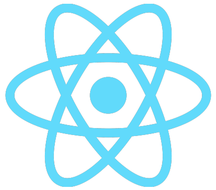
Google Style Guides

Every major open-source project has its
project. It is much easier to understand a

"Style" covers a lot of ground, from "use
project (google/styleguide) links to the st
you may be pointed to this page to see th

- AngularJS Style Guide
- Common Lisp Style Guide

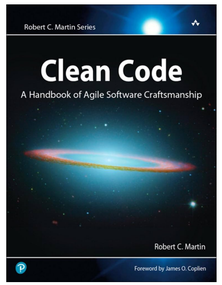
👷 Concepteurs de langages ou de bibliothèques

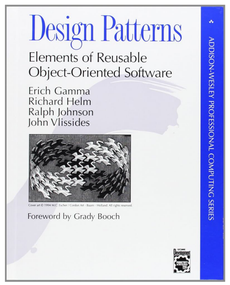


Les Pratiques de Code

Origines

📚 Livres





⌨️ Blogs

The Clean Code Blog

by Robert C. Martin (Uncle Bob)

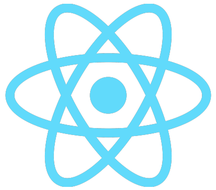
Google Style Guides

Every major open-source project has its
project. It is much easier to understand a

"Style" covers a lot of ground, from "use
project (google/styleguide) links to the st
you may be pointed to this page to see th

- AngularJS Style Guide
- Common Lisp Style Guide

👷 Concepteurs de langages ou de bibliothèques

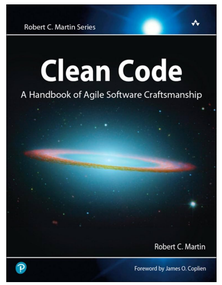


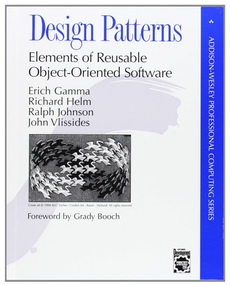
🛠️ Outils

Les Pratiques de Code

Origines

📚 Livres





⌨️ Blogs

The Clean Code Blog

by Robert C. Martin (Uncle Bob)

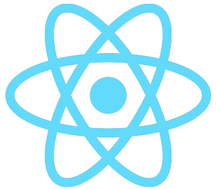
Google Style Guides

Every major open-source project has its
project. It is much easier to understand a

"Style" covers a lot of ground, from "use
project (google/styleguide) links to the st
you may be pointed to this page to see th

- AngularJS Style Guide
- Common Lisp Style Guide

👷 Concepteurs de langages ou de bibliothèques



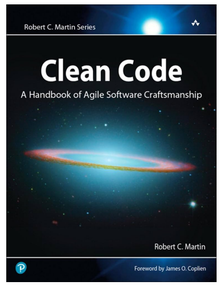
🛠️ Outils

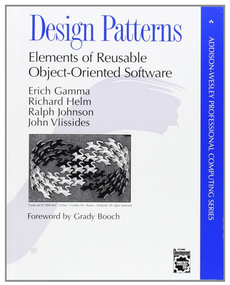
👷‍💻 Pratiques internes

Les Pratiques de Code

Origines

📚 Livres





⌨️ Blogs

The Clean Code Blog

by Robert C. Martin (Uncle Bob)

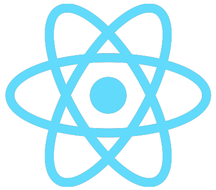
Google Style Guides

Every major open-source project has its
project. It is much easier to understand a

"Style" covers a lot of ground, from "use
project (google/styleguide) links to the st
you may be pointed to this page to see th

- AngularJS Style Guide
- Common Lisp Style Guide

👷 Concepteurs de langages ou de bibliothèques



🛠️ Outils

👨‍💻 Pratiques internes

Les Pratiques de Code

Diffusion - Revue de Code

Les Pratiques de Code

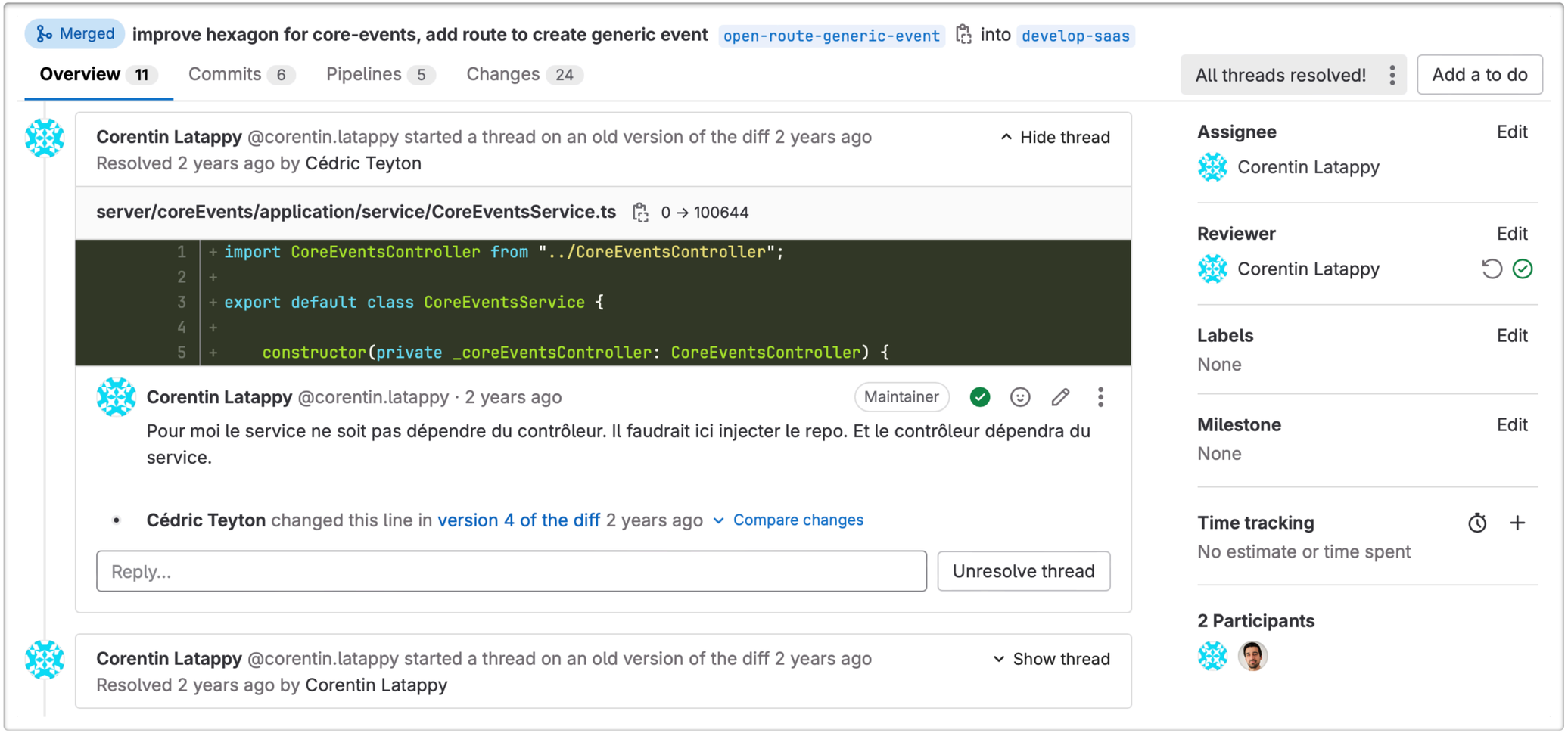
Diffusion - Revue de Code

Système de revue par un pair

Les Pratiques de Code

Diffusion - Revue de Code

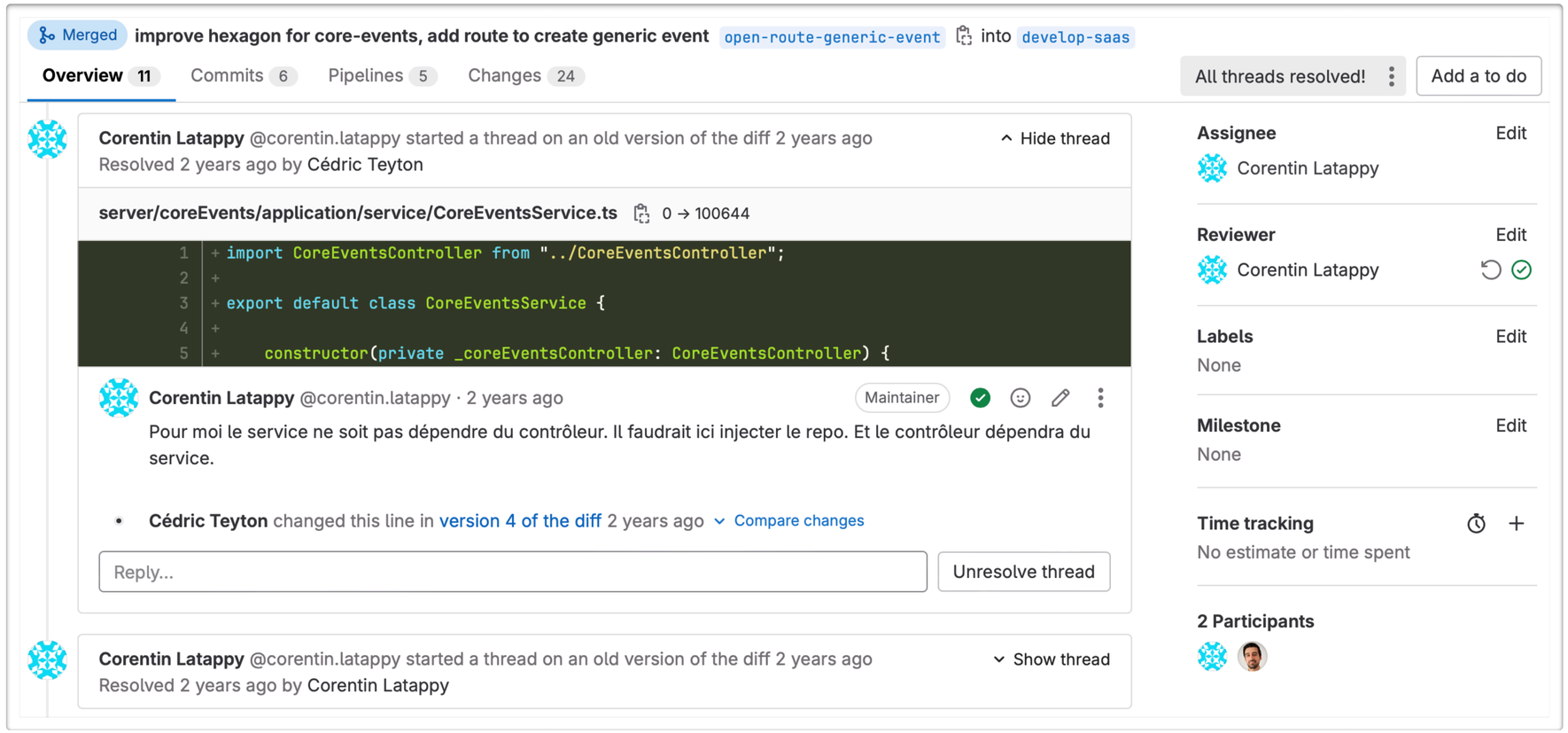
Système de revue par un pair



Les Pratiques de Code

Diffusion - Revue de Code

Système de revue par un pair

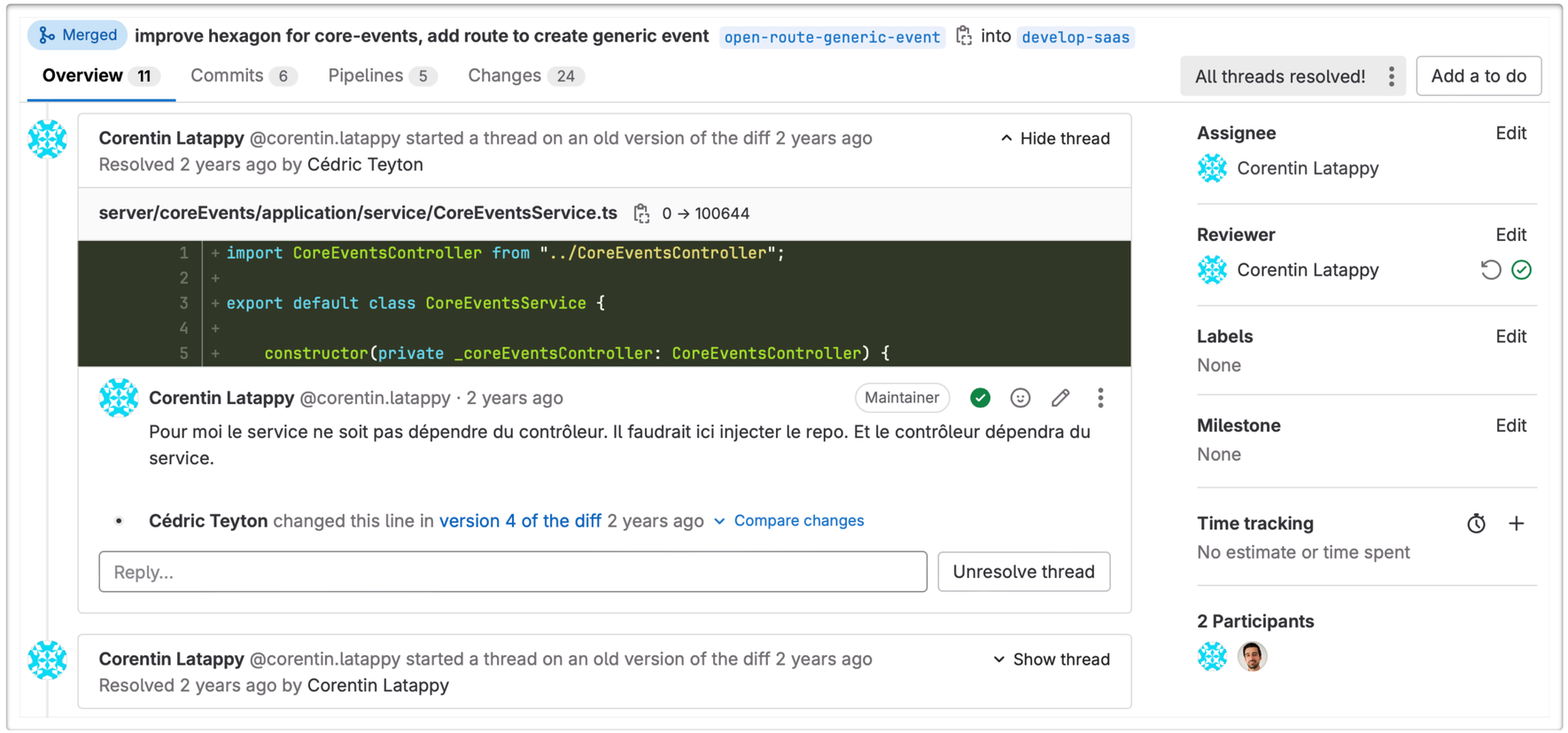


Temps

Les Pratiques de Code

Diffusion - Revue de Code

Système de revue par un pair



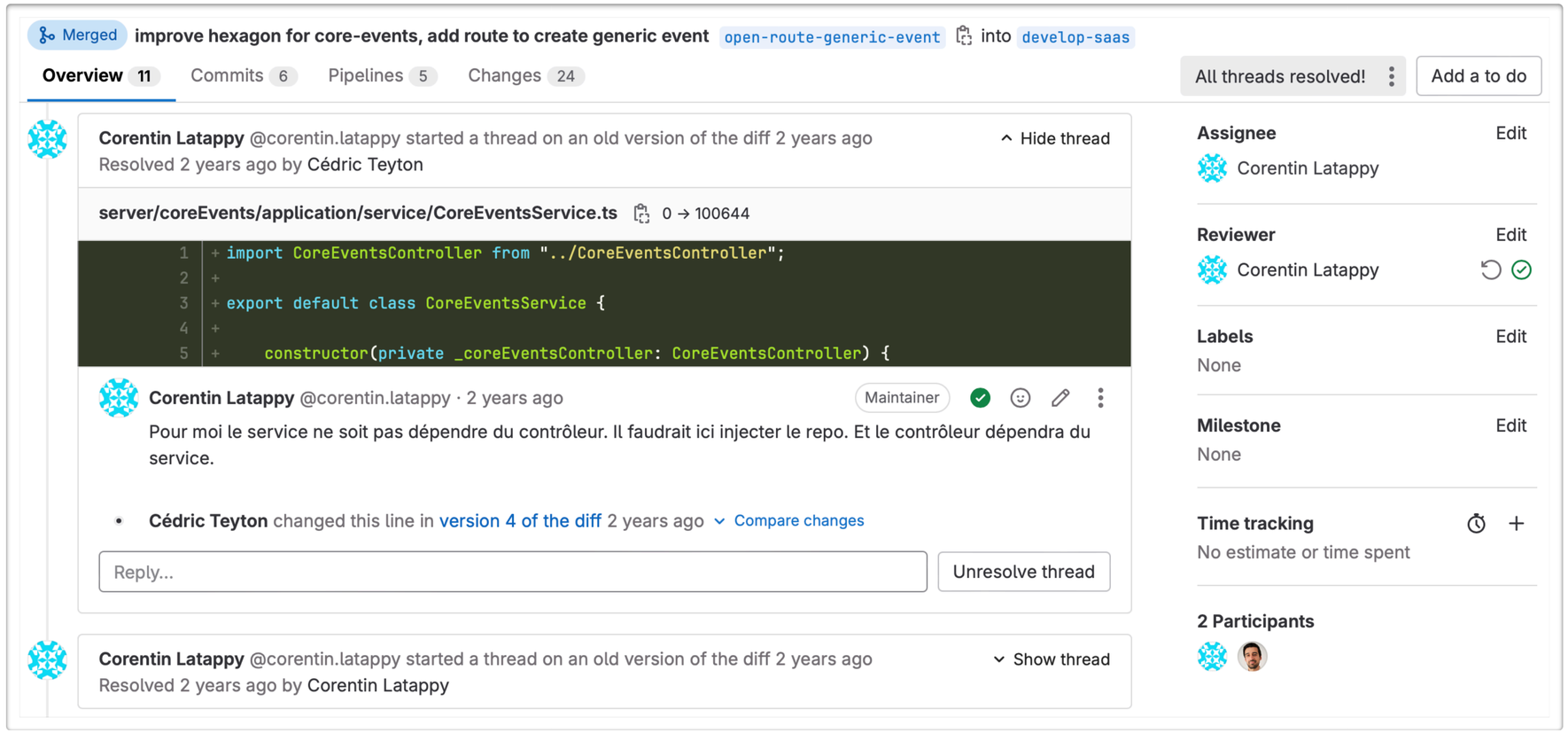
Temps

Impactant

Les Pratiques de Code

Diffusion - Revue de Code

Système de revue par un pair



Temps

Impactant

Manuelle

Les Pratiques de Code

Diffusion - Outils d'Analyse Statique, *aka* Linters

Les Pratiques de Code

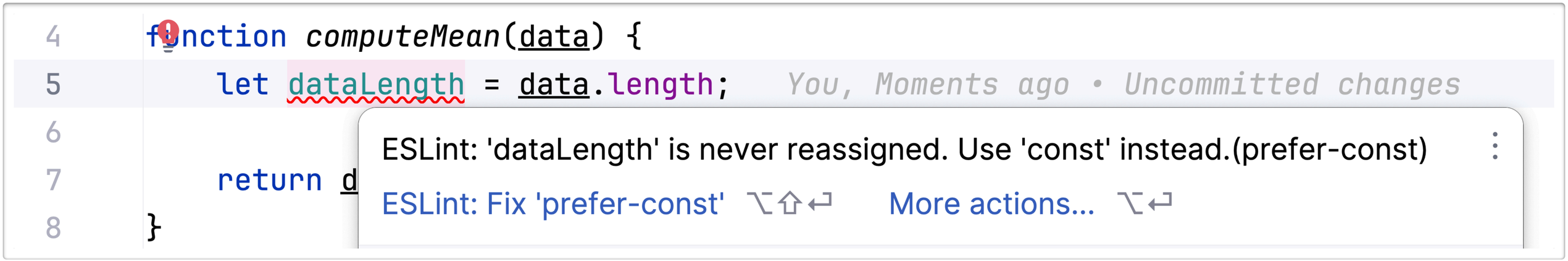
Diffusion - Outils d'Analyse Statique, *aka* Linters

Analyse automatique du code

Les Pratiques de Code

Diffusion - Outils d'Analyse Statique, *aka* Linters

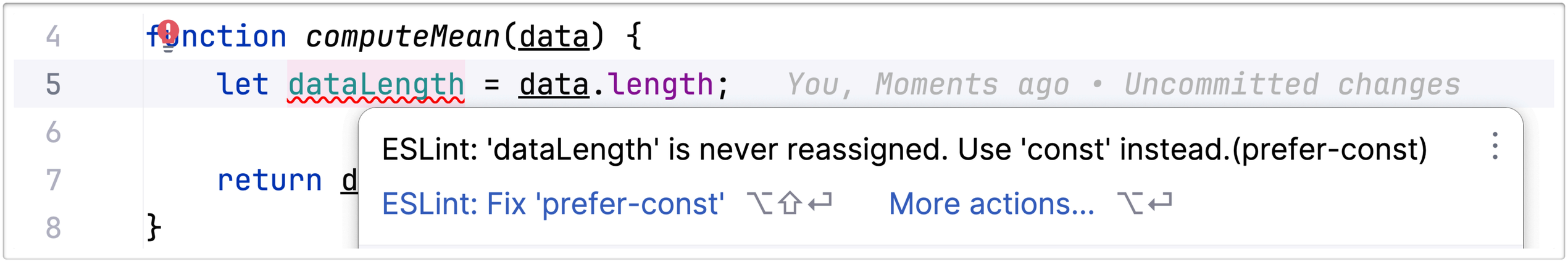
Analyse automatique du code



Les Pratiques de Code

Diffusion - Outils d'Analyse Statique, *aka* Linters

Analyse automatique du code

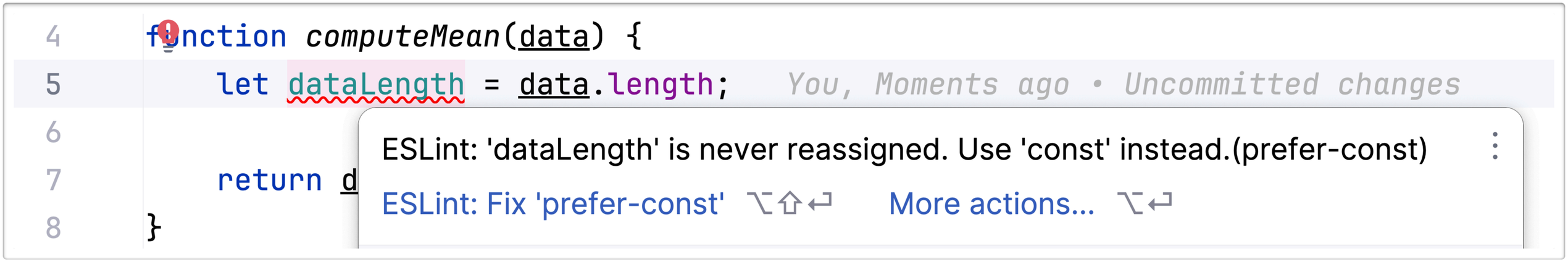


Activation

Les Pratiques de Code

Diffusion - Outils d'Analyse Statique, *aka* Linters

Analyse automatique du code



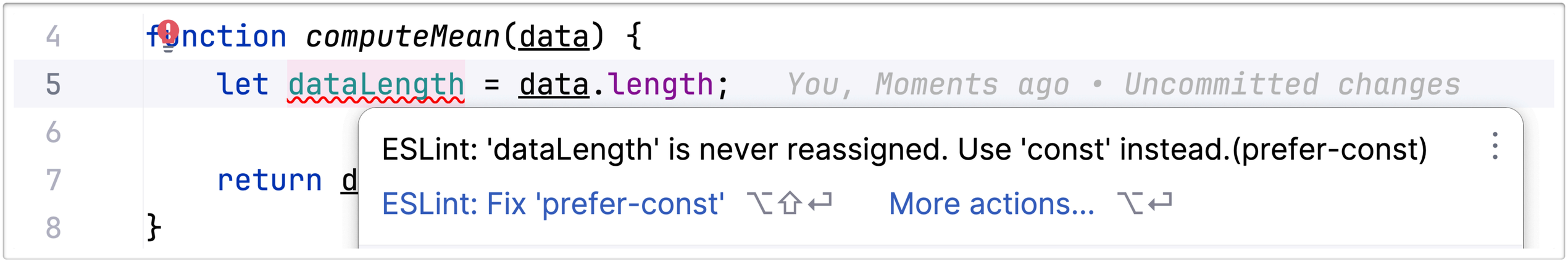
Activation

Règles

Les Pratiques de Code

Diffusion - Outils d'Analyse Statique, *aka* Linters

Analyse automatique du code



Activation Règles Faux-positifs

Les Pratiques de Code

La Vision Packmind

Les Pratiques de Code

La Vision Packmind

Outil permettant la mise en commun des pratiques internes

Les Pratiques de Code

La Vision Packmind

Outil permettant la mise en commun des pratiques internes

```
5   const fetchAllData = async (): Promise<AllData> => {
6       const rawResultRes : Response  = await fetch( input: '/data/results.json');
7       const rawResult: RawResult = await rawResultRes.json();
8
9       const coding : {file: string, name: string}[]  = [
10          { file: '/data/open_coding/what.json', name: 'What' },
11          { file: '/data/open_coding/why.json', name: 'Why' },
12          { file: '/data/open_coding/fix.json', name: 'Fix' },
13          { file: '/data/open_coding/other.json', name: 'other' },
14          { file: '/data/open_coding/general.json', name: 'general' },
15      ];
16
17      const codings = await Promise.all(coding.map(async ({ file : string,  name : string  }) : Promise<
```

Les Pratiques de Code

La Vision Packmind

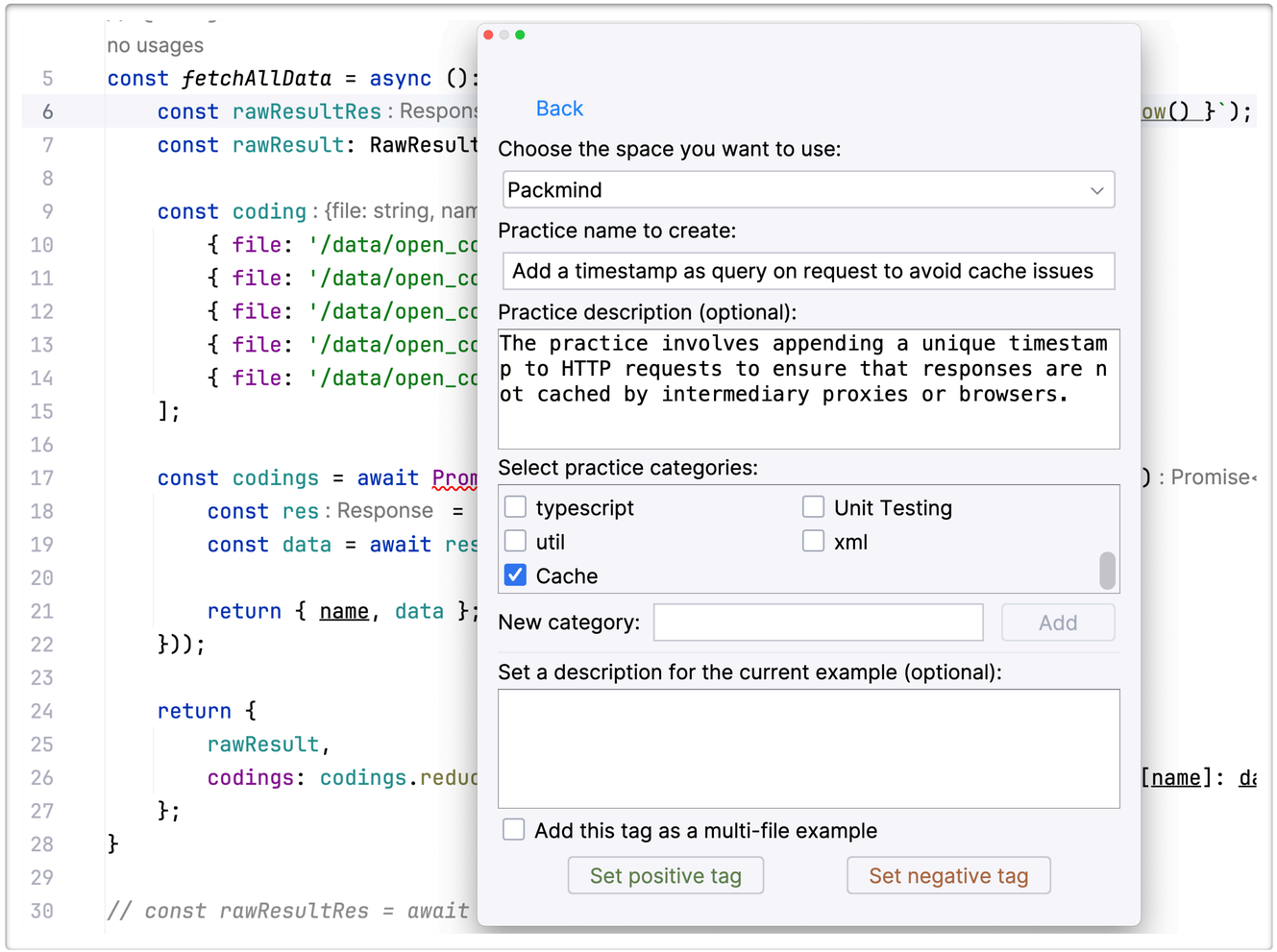
Outil permettant la mise en commun des pratiques internes

```
5    const fetchAllData = async (): Promise<AllData> => {
6        const rawResultRes : Response  = await fetch( input: `/data/results.json?t=${ Date.now() }`);
7        const rawResult: RawResult = await rawResultRes.json();
8
9        const coding : {file: string, name: string}[]  = [
10           { file: '/data/open_coding/what.json', name: 'What' },
11           { file: '/data/open_coding/why.json', name: 'Why' },
12           { file: '/data/open_coding/fix.json', name: 'Fix' },
13           { file: '/data/open_coding/other.json', name: 'other' },
14           { file: '/data/open_coding/general.json', name: 'general' },
15       ];
16
17       const codings = await Promise.all(coding.map(async ({ file : string,  name : string  }) : Promise<
```

Les Pratiques de Code

La Vision Packmind

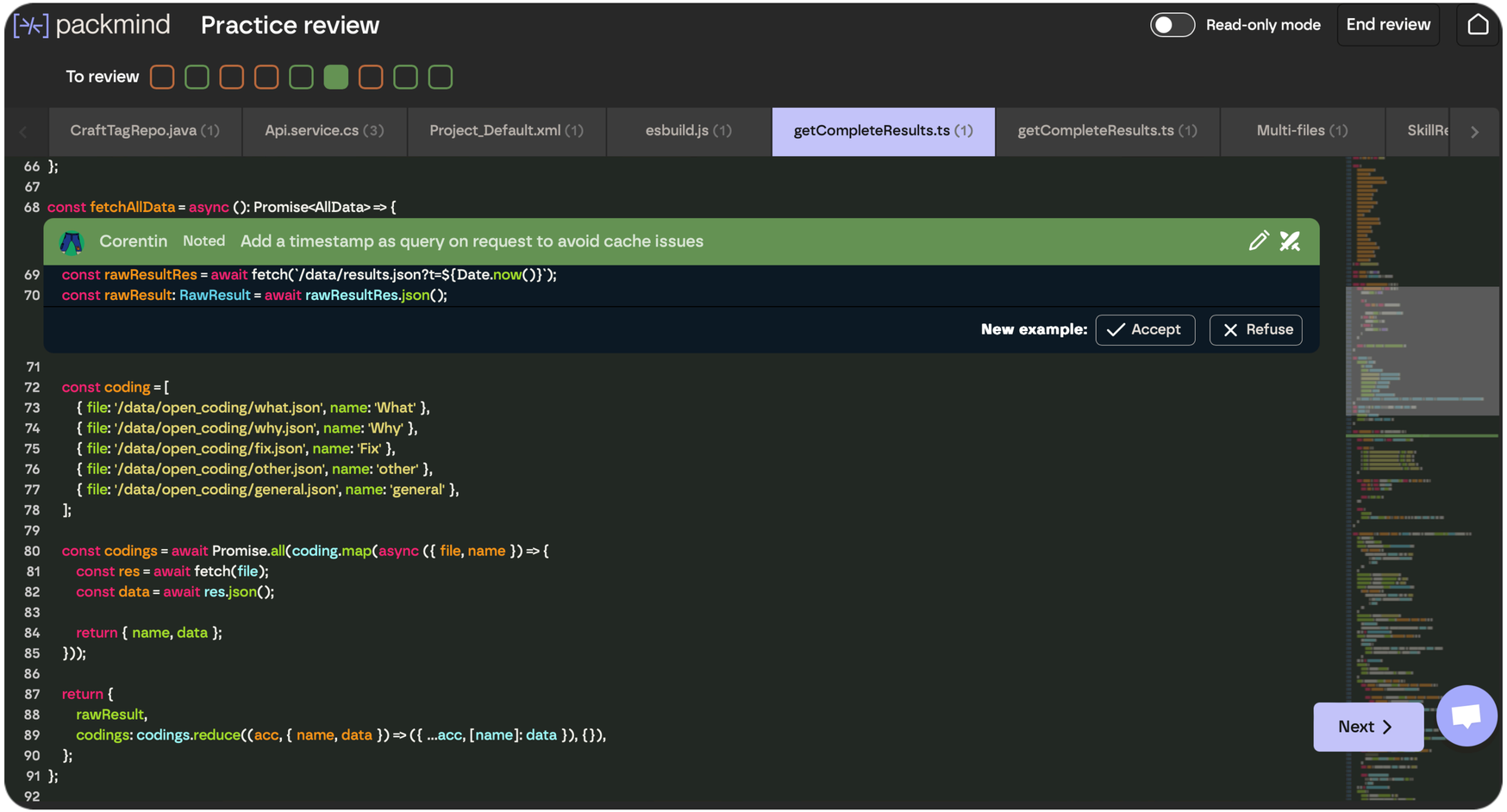
Outil permettant la mise en commun des pratiques internes



Les Pratiques de Code

La Vision Packmind

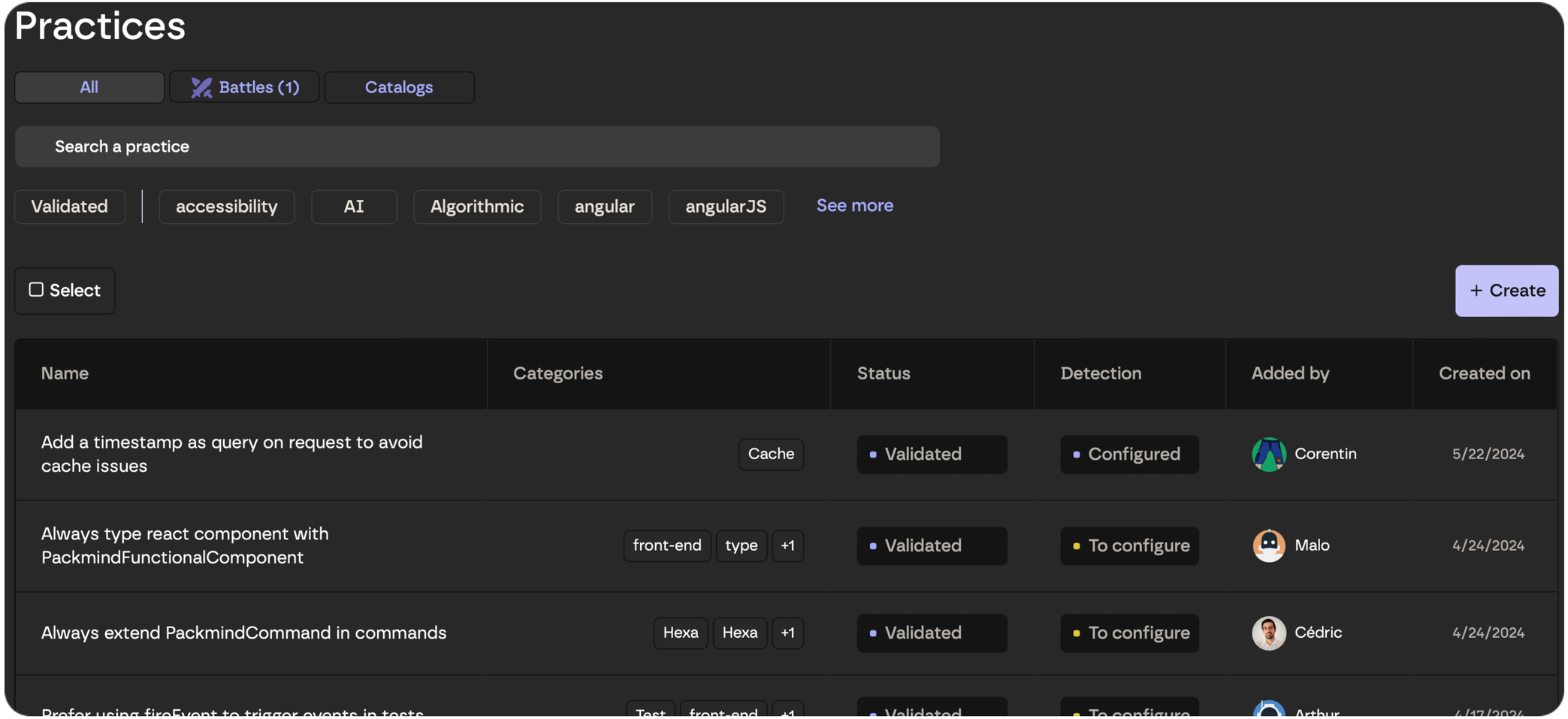
Outil permettant la mise en commun des pratiques internes



Les Pratiques de Code

La Vision Packmind

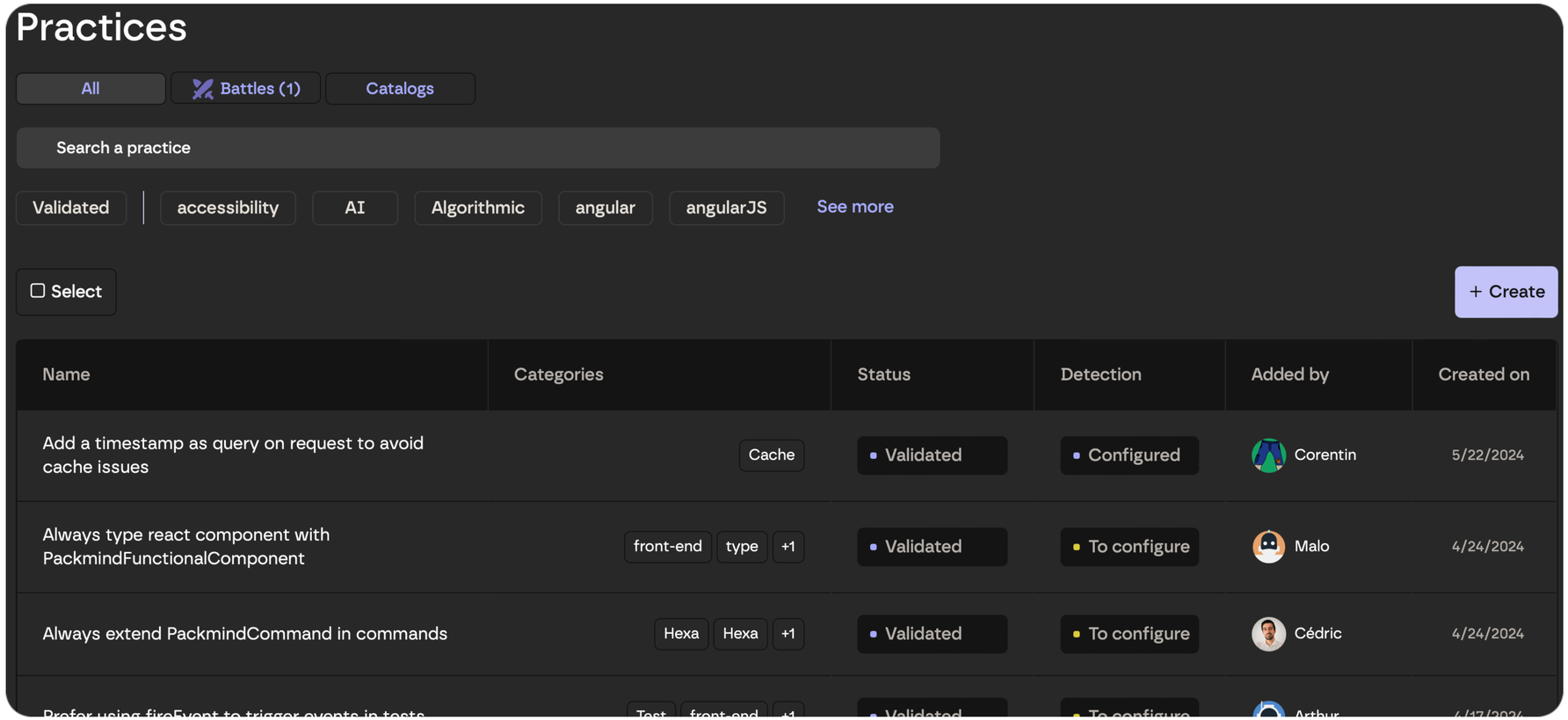
Outil permettant la mise en commun des pratiques internes



Les Pratiques de Code

La Vision Packmind

Outil permettant la mise en commun des pratiques internes

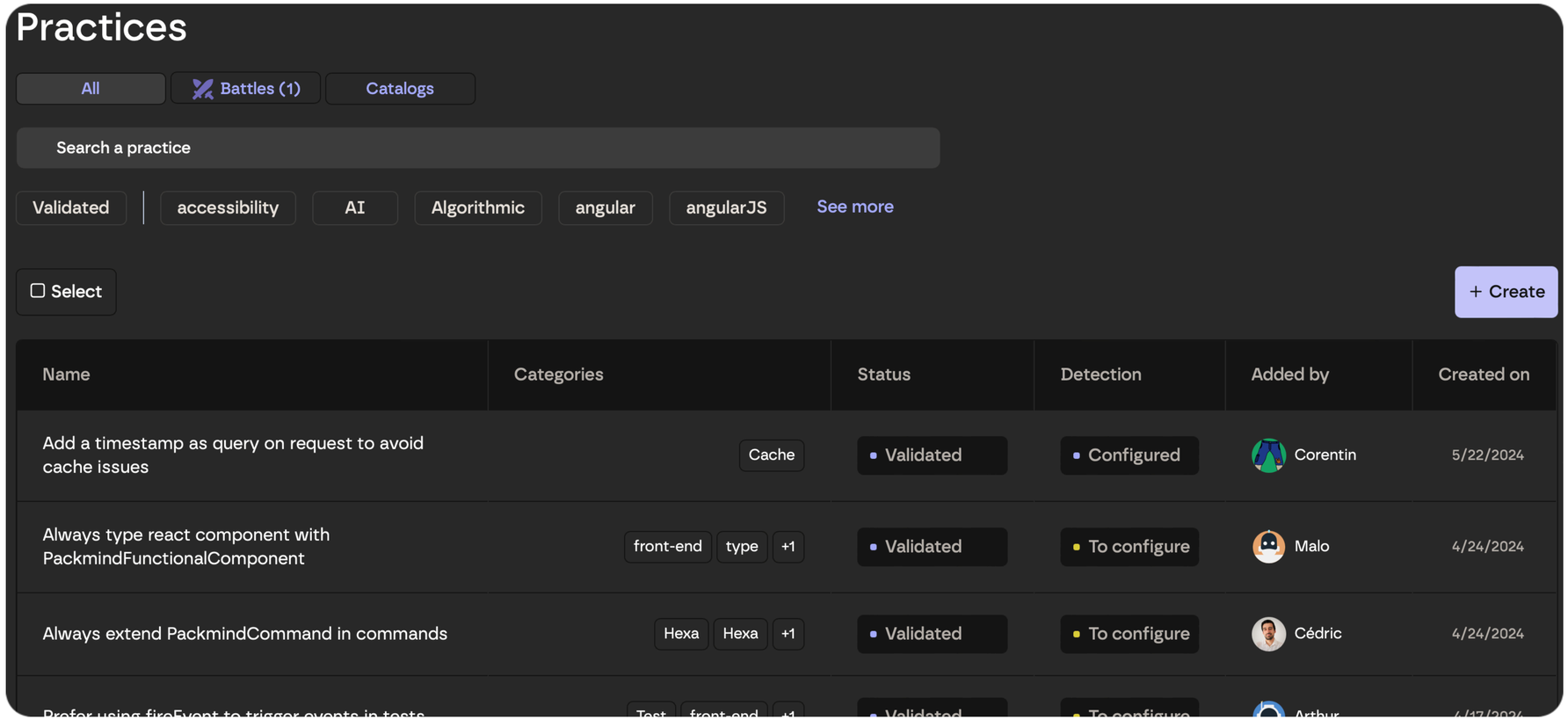


Documentation

Les Pratiques de Code

La Vision Packmind

Outil permettant la mise en commun des pratiques internes



Documentation

Détection

Objectifs de cette thèse

Objectifs de cette thèse

Améliorer la qualité de la documentation des pratiques

Objectifs de cette thèse

Améliorer la qualité de la documentation des pratiques

Détecter les violations de pratiques internes

Les Pratiques de Code : De la Documentation à la Détection

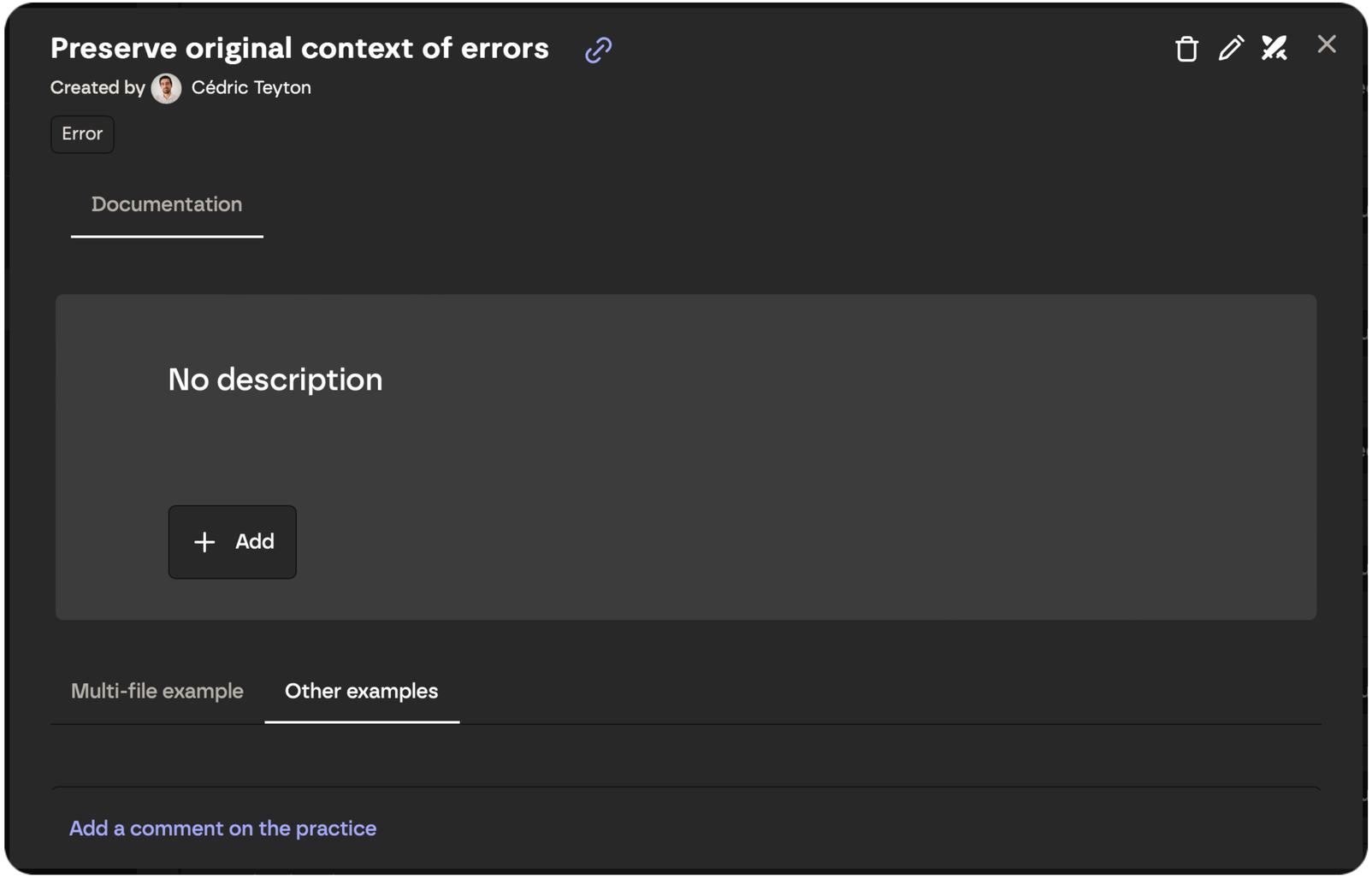
C. Latappy, T. Degueule, J.-R. Falleri, R. Robbes, X. Blanc, et C. Teyton,
« What the Fix? A Study of ASAT Rules Documentation »
in *Proceedings of the 32nd IEEE/ACM International Conference on Program Comprehension, ICPC 2024*
doi: 10.1145/3643916.3644404.

[-/-] packmind

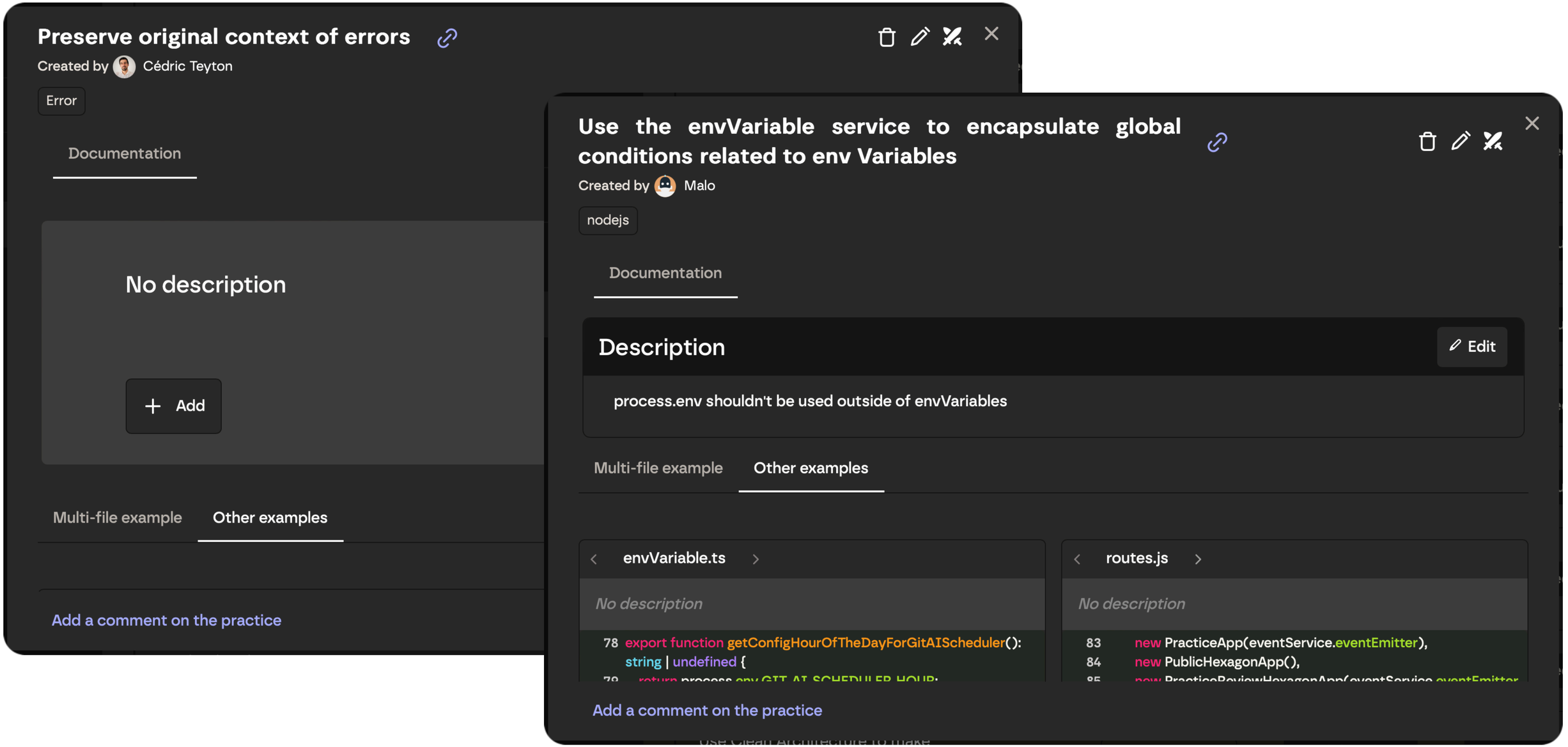
LaBRI

Chez [⋋] packmind

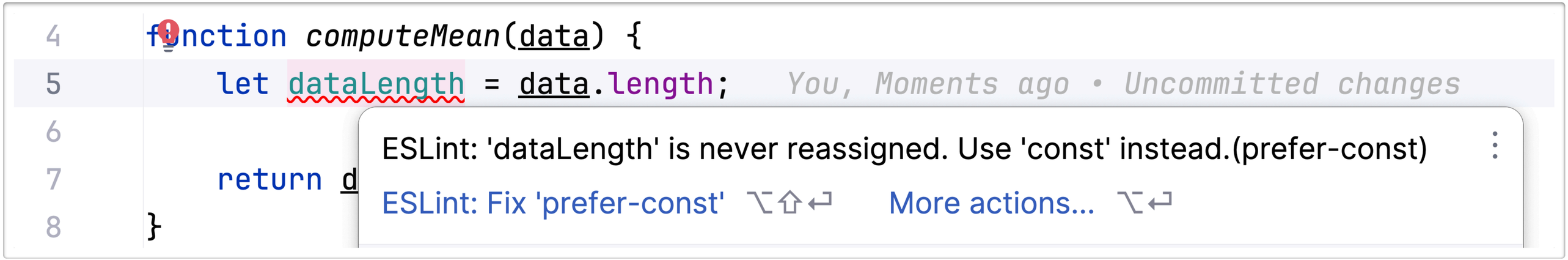
Chez packmind



Chez packmind



Chez d’autres outils existants



Règle *prefer-const* provenant de *ESLint* sur du code *JavaScript*

prefer-const

Require `const` declarations for variables that are never reassigned after declared

🔧 Some problems reported by this rule are automatically fixable by the `--fix` command line option

If a variable is never reassigned, using the `const` declaration is better.

`const` declaration tells readers, “this variable is never reassigned,” reducing cognitive load and improving maintainability.

Rule Details

This rule is aimed at flagging variables that are declared using `let` keyword, but never reassigned after the initial assignment.

Examples of **incorrect** code for this rule:

```
/*eslint prefer-const: "error"*/

// it's initialized and never reassigned.
let a = 3;
console.log(a);

// `i` is redefined (not reassigned) on each loop step.
for (let i in [1, 2, 3]) {
    console.log(i);
}
```

Open in Playground

Examples of **correct** code for this rule:

```
/*eslint prefer-const: "error"*/

// using const.
const a = 0;

// `i` gets a new binding each iteration
for (const i in [1, 2, 3]) {
  console.log(i);
}
```

Open in Playground

Options

```
{
    "prefer-const": ["error", {
        "destructuring": "any",
        "ignoreReadBeforeAssign": false
    }]
}
```

When Not To Use It

If you don’t want to be notified about variables that are never reassigned after initial assignment, you can safely disable this rule.

Related Rules

no-var → no-use-before-define →

Version

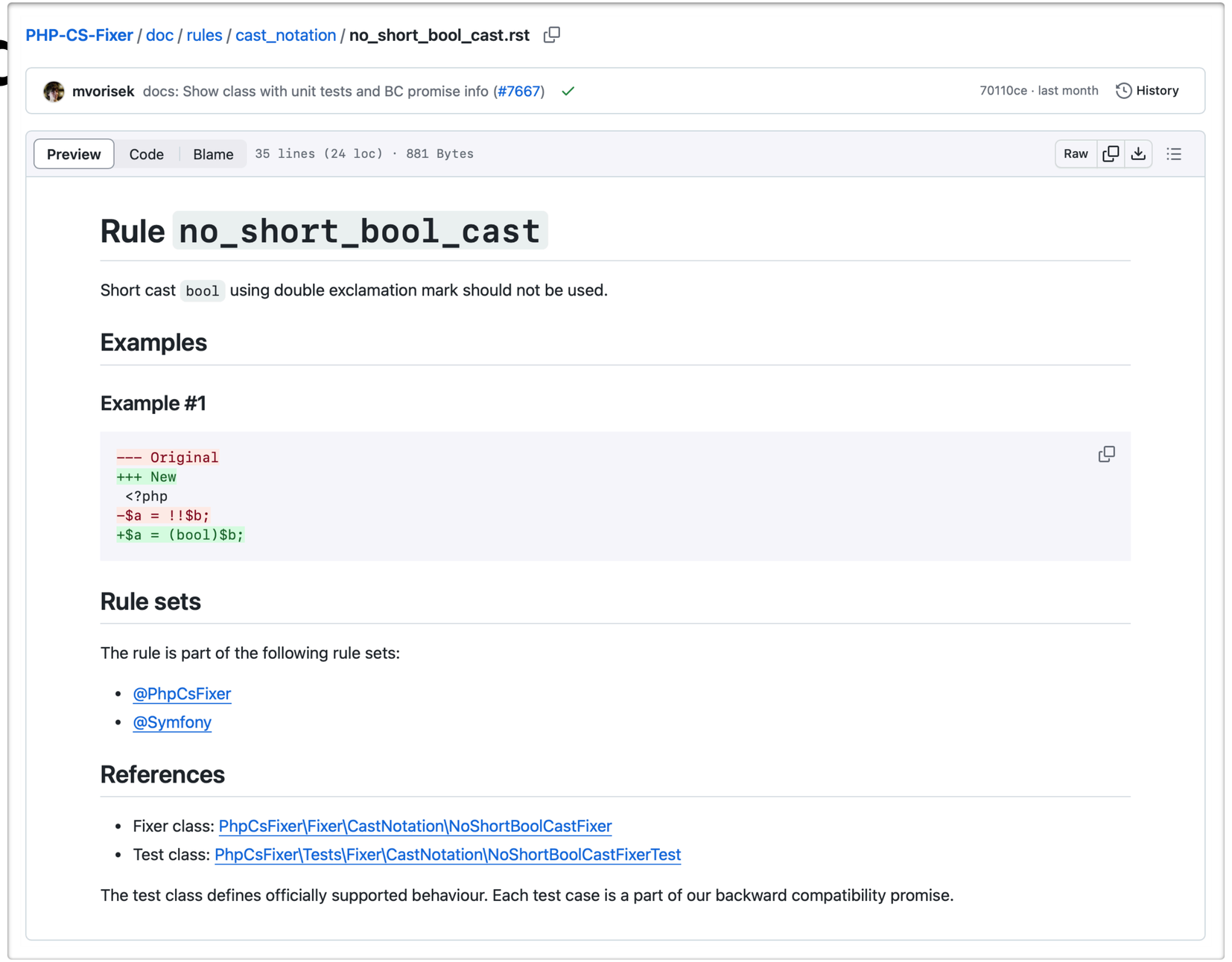
This rule was introduced in ESLint v0.23.0.

Resources

- Rule source
- Tests source

Règle *prefer-const* de *ESLint*

Chez d



PHP-CS-Fixer / doc / rules / cast_notation / no_short_bool_cast.rst

mvorisek docs: Show class with unit tests and BC promise info (#7667) ✓ 70110ce · last month History

Preview | Code | Blame 35 lines (24 loc) · 881 Bytes Raw

Rule `no_short_bool_cast`

Short cast `bool` using double exclamation mark should not be used.

Examples

Example #1

```
--- Original
+++ New
 <?php
-$a = !!$b;
+$a = (bool)$b;
```

Rule sets

The rule is part of the following rule sets:

- @PhpCsFixer
- @Symfony

References

- Fixer class: PhpCsFixer\Fixer\CastNotation\NoShortBoolCastFixer
- Test class: PhpCsFixer\Tests\Fixer\CastNotation\NoShortBoolCastFixerTest

The test class defines officially supported behaviour. Each test case is a part of our backward compatibility promise.

Règle *no_short_bool_cast* de *PHP CS Fixer*

prefer-const

Require `const` declarations for variables that are never reassigned after declared

Some problems reported by this rule are automatically fixable by the `--fix` command line option

If a variable is never reassigned, using the `const` declaration is better.

`const` declaration tells readers, "this variable is never reassigned," reducing cognitive load and improving maintainability.

Rule Details

This rule is aimed at flagging variables that are declared using `let` keyword, but never reassigned after the initial assignment.

Examples of **incorrect** code for this rule:

```
/*eslint prefer-const: "error"*/

// it's initialized and never reassigned.
let a = 3;
console.log(a);

// `i` is redefined (not reassigned) on each loop step.
for (let i in [1, 2, 3]) {
    console.log(i);
}
```

Open in Playground

Examples of **correct** code for this rule:

```
/*eslint prefer-const: "error"*/

// using const.
const a = 0;

// `i` gets a new binding each iteration
for (const i in [1, 2, 3]) {
  console.log(i);
}
```

Options

```
{
    "prefer-const": ["error", {
        "destructuring": "any",
        "ignoreReadBeforeAssign": false
    }]
}
```

When Not To Use It

If you don't want to be notified about variables that are never reassig

Related Rules

no-var → no-use-before-define →

Version

This rule was introduced in ESLint v0.23.0.

Resources

- Rule source
- Tests source

SQL Injection

Injection is #1 on the 2010 OWASP Top Ten web security risks. SQL injection is wh
able to manipulate a value which is used unsafely inside a SQL query. This can lea
data loss, elevation of privilege, and other unpleasant outcomes.

Brakeman focuses on ActiveRecord methods dealing with building SQL statements

A basic (Rails 2.x) example looks like this:

```
User.first(:conditions => "username = '#{params[:username]}'")
```

Brakeman would produce a warning like this:

```
Possible SQL injection near line 30: User.first(:conditions => ("username = '#{param
```

The safe way to do this query is to use a parameterized query:

```
User.first(:conditions => ["username = ?", params[:username]])
```

Brakeman also understands the new Rails 3.x way of doing things (and local variab
concatentation):

```
username = params[:user][:name].downcase
password = params[:user][:password]

User.first.where("username = '" + username + "' AND password = '" + password + "'")
```

This results in this kind of warning:

```
Possible SQL injection near line 37:
User.first.where(((((("username = '" + params[:user][:name].downcase) + "' AND passwo
```

See the Ruby Security Guide for more information and Rails-SQLi.org for many ex
injection in Rails.

MultipleVariableDeclarations

Since Checkstyle 3.4

Description

Checks that each variable declaration is in its own statement and on its own line.

Rationale: the Java code conventions chapter 6.1 recommends that declarations should be one per line/statement.

Examples

To configure the check:

```
<module name="Checker">
  <module name="TreeWalker">
    <module name="MultipleVariableDeclarations"/>
  </module>
</module>
```

Example:

```
public class Test {
  public void myTest() {
    int mid;
    int high;
    // ...

    int lower, higher; // violation
    // ...

    int value,
        index; // violation
    // ...

    int place = mid, number = high;  // violation
  }
}
```

Example Of Usage

- Google Style
- Sun Style
- Checkstyle Style

Violation Messages

- multiple.variable.declarations
- multiple.variable.declarations.comma

All messages can be customized if the default message doesn't suit you. Pl
how to.

Package

com.puppycrawl.tools.checkstyle.checks.coding

consider-using-generator / R1728

Message emitted:

`Consider using a generator instead '%s(%s)'`

Description:

If your container can be large using a generator will bring better performance.

Problematic code:

```
list([0 for y in list(range(10))])  # [consider-using-generator]
tuple([0 for y in list(range(10))])  # [consider-using-generator]
sum([y**2 for y in list(range(10))])  # [consider-using-generator]
max([y**2 for y in list(range(10))])  # [consider-using-generator]
min([y**2 for y in list(range(10))])  # [consider-using-generator]
```

Correct code:

```
list(0 for y in list(range(10)))
tuple(0 for y in list(range(10)))
sum(y**2 for y in list(range(10)))
max(y**2 for y in list(range(10)))
min(y**2 for y in list(range(10)))
```

Additional details:

Removing `[]` inside calls that can use containers or generators should be considered for performance reasons since a generator will have an upfront cost to pay. The performance will be better if you are working with long lists or sets.

For `max`, `min` and `sum` using a generator is also recommended by pep289.

Related links:

- PEP 289

postfixOperator - CWE 398

对于非基本类型来说，使用前缀++ / --更好，后缀是低效率的

Type

id = "postfixOperator"
severity = "performance"

cwe = "CWE-398"
cwe-type = "Variant"

```
<error id="postfixOperator" severity="performance" msg="Prefer prefix ++/-- operators for non-primitive types."
```

Description

Prefix ++/-- operators should be preferred for non-primitive types. Pre-increment/decrement can be more efficient than post-increment/decrement. Post-increment/decrement usually involves keeping a copy of the previous value around and adds a little code.

Example cpp file

```
void main(){
for(vector<int>::iterator iter=vector_database.begin(); vector_database!=veci.end(); iter++)
        if( *iter == 10)
                vector_database.erase(iter);

}
```

Massage output in cppcheck

```
[test.cpp:2]: (performance) Prefer prefix ++/-- operators for non-primitive types.
```

XML output cppcheck

```
<?xml version="1.0" encoding="UTF-8"?>
<results version="2">
    <cppcheck version="1.83"/>
    <errors>
        <error id="postfixOperator" severity="performance" msg="Prefer prefix ++/-- operators for non-primitive t
            <location file="test.cpp" line="2"/>
        </error>
    </errors>
</results>
```

CompareWithEmptyString

This rule will fire if a string is com
possibility (with .NET 2.0) is to us

Bad example:

```
public void SimpleMethod
{
    if (myString.Equals
    }
}
```

Good example:

```
public void SimpleMethod (string myString)
{
    if (
    }
}
```

Function is too complex (C901)

Functions that are deemed too complex are functions that have too much branching logic. Branching logic includes `if` / `elif` / `else` and `for` / `while` loops.

Anti-pattern

The following example has a complexity score of 5, because there are five potential branches.

```
def post_comment(self):
    if self.success:
        comment = 'Build succeeded'
    elif self.warning:
        comment = 'Build had issues'
    elif self.failed:
        comment = 'Build failed'

    if self.success:
        self.post(comment, type='success')
    else:
        self.post(comment, type='error')
```

Best practice

ForLoopShouldBeWhileLoop

Since: 0.6

Name: for loop should be while loop

Under certain circumstances, some `for` loops can be simplified to `while` loops to make code more concise.

This rule is defined by the following class: oclint-rules/rules/basic/ForLoopShouldBeWhileLoopRule.cpp

Example:

```
void example(int a)
{
    for (; a < 100;)
    {
        foo(a);
    }
}
```

Preview Code

Rule

Short cas

Examples

Example #1

```
--- Original
+++ New
 <?php
-$a = !!$b;
+$a = (bool)$b;
```

Rule sets

The rule is part of the following rule sets:

- @PhpCsFixer
- @Symfony

References

- Fixer class: PhpCsFixer\Fixer\CastNotation\NoShortBoolCastFixer
- Test class: PhpCsFixer\Tests\Fixer\CastNotation\NoShortBoolCastFixerTest

The test class defines officially supported behaviour. Each test case is a part of our backward compatibility promise.

Dans l’État de l’Art

Dans l’État de l’Art

Outils d’Analyse Statique

Dans l’État de l’Art

Outils d’Analyse Statique

Taxonomie sur les Outils d’Analyse Statique

Novak *et al.* 2010

Dans l’État de l’Art

Outils d’Analyse Statique

Taxonomie sur les Outils d’Analyse Statique

Novak *et al.* 2010

Exploration de 4 outils pour 2 langages de programmation

Dans l’État de l’Art

Outils d’Analyse Statique

Taxonomie sur les Outils d’Analyse Statique

Novak *et al.* 2010

- Exploration de 4 outils pour 2 langages de programmation
- 10 catégories (Configuration, Technologie, Types, …)

Dans l’État de l’Art

Outils d’Analyse Statique

Taxonomie sur les Outils d’Analyse Statique — Novak *et al.* 2010

- Exploration de 4 outils pour 2 langages de programmation
- 10 catégories (Configuration, Technologie, Types, …)

Enquêtes sur la qualité perçue des notifications — Johnson *et al.* 2013, Tahaei *et al.* 2021

Dans l’État de l’Art

Outils d’Analyse Statique

Taxonomie sur les Outils d’Analyse Statique — Novak *et al.* 2010

- Exploration de 4 outils pour 2 langages de programmation
- 10 catégories (Configuration, Technologie, Types, …)

Enquêtes sur la qualité perçue des notifications — Johnson *et al.* 2013, Tahaei *et al.* 2021

- Interview / Questionnaire auprès de développeurs

Dans l’État de l’Art

Outils d’Analyse Statique

Taxonomie sur les Outils d’Analyse Statique — Novak *et al.* 2010

- Exploration de 4 outils pour 2 langages de programmation
- 10 catégories (Configuration, Technologie, Types, …)

Enquêtes sur la qualité perçue des notifications — Johnson *et al.* 2013, Tahaei *et al.* 2021

- Interview / Questionnaire auprès de développeurs
- Manque d’informations fournies

Dans l’État de l’Art

Outils d’Analyse Statique

Taxonomie sur les Outils d’Analyse Statique — Novak *et al.* 2010

- Exploration de 4 outils pour 2 langages de programmation
- 10 catégories (Configuration, Technologie, Types, …)

Enquêtes sur la qualité perçue des notifications — Johnson *et al.* 2013, Tahaei *et al.* 2021

- Interview / Questionnaire auprès de développeurs
- Manque d’informations fournies

Documentation logicielle

Dans l'État de l'Art

Outils d'Analyse Statique

Taxonomie sur les Outils d'Analyse Statique — Novak *et al.* 2010

- Exploration de 4 outils pour 2 langages de programmation
- 10 catégories (Configuration, Technologie, Types, …)

Enquêtes sur la qualité perçue des notifications — Johnson *et al.* 2013, Tahaei *et al.* 2021

- Interview / Questionnaire auprès de développeurs
- Manque d'informations fournies

Documentation logicielle

Étude sur les problématiques rencontrées par les développeurs — Aghajani *et al.* 2019

Dans l’État de l’Art

Outils d’Analyse Statique

Taxonomie sur les Outils d’Analyse Statique — Novak *et al.* 2010

- Exploration de 4 outils pour 2 langages de programmation
- 10 catégories (Configuration, Technologie, Types, …)

Enquêtes sur la qualité perçue des notifications — Johnson *et al.* 2013, Tahaei *et al.* 2021

- Interview / Questionnaire auprès de développeurs
- Manque d’informations fournies

Documentation logicielle

Étude sur les problématiques rencontrées par les développeurs — Aghajani *et al.* 2019

- Création d’une taxonomie à partir de 3 sources différentes

Dans l'État de l'Art

Outils d'Analyse Statique

Taxonomie sur les Outils d'Analyse Statique — Novak *et al.* 2010

- Exploration de 4 outils pour 2 langages de programmation
- 10 catégories (Configuration, Technologie, Types, …)

Enquêtes sur la qualité perçue des notifications — Johnson *et al.* 2013, Tahaei *et al.* 2021

- Interview / Questionnaire auprès de développeurs
- Manque d'informations fournies

Documentation logicielle

Étude sur les problématiques rencontrées par les développeurs — Aghajani *et al.* 2019

- Création d'une taxonomie à partir de 3 sources différentes
- 4 catégories (Contenu, Forme, …)

Question de Recherche

Question de Recherche

Qu’est-ce qu’une bonne documentation pour une pratique de code ?

Analyse de la Documentation

Analyse de la Documentation

no-var

Require `let` or `const` instead of `var`

ECMAScript 6 allows programmers to create variables with block scope instead of function scope using the `let` and `const` keywords. Block scope is common in many other programming languages and helps programmers avoid mistakes such as:

```
1   var count = people.length;
2   var enoughFood = count > sandwiches.length;
3
4   if (enoughFood) {
5       var count = sandwiches.length; // accidentally overriding the cou
6       console.log("We have " + count + " sandwiches for everyone. Plent
7   }
8
9   // our count variable is no longer accurate
10  console.log("We have " + count + " people and " + sandwiches.length +
```

Rule Details

This rule is aimed at discouraging the use of `var` and encouraging the use of `const` or `let` instead.

Examples

Examples of **incorrect** code for this rule:

```
1   /*eslint no-var: "error"*/
2
3   var x = "y";
4   var CONFIG = {};
```

Open in Playground

Examples of **correct** code for this rule:

```
1   /*eslint no-var: "error"*/
2   /*eslint-env es6*/
3
4   let x = "y";
5   const CONFIG = {};
```

Open in Playground

When Not To Use It

In addition to non-ES6 environments, existing JavaScript projects that are beginning to introduce ES6 into their codebase may not want to apply this rule if the cost of migrating from `var` to `let` is too costly.

Version

This rule was introduced in ESLint v0.12.0.

Resources

- Rule source
- Tests source

Analyse de la Documentation

no-var

Nom

Require `let` or `const` instead of `var`

Description

ECMAScript 6 allows programmers to create variables with block scope instead of function scope using the `let` and `const` keywords. Block scope is common in many other programming languages and helps programmers avoid mistakes such as:

```
var count = people.length;
var enoughFood = count > sandwiches.length;

if (enoughFood) {
    var count = sandwiches.length; // accidentally overriding the cou
    console.log("We have " + count + " sandwiches for everyone. Plent
}

// our count variable is no longer accurate
console.log("We have " + count + " people and " + sandwiches.length +
```

Rule Details

This rule is aimed at discouraging the use of `var` and encouraging the use of `const` or `let` instead.

Examples

Examples of **incorrect** code for this rule:

Exemple négatif

```
/*eslint no-var: "error"*/

var x = "y";
var CONFIG = {};
```

Open in Playground

Examples of **correct** code for this rule:

Exemple positif

```
/*eslint no-var: "error"*/
/*eslint-env es6*/

let x = "y";
const CONFIG = {};
```

Open in Playground

When Not To Use It

Avertissements

In addition to non-ES6 environments, existing JavaScript projects that are beginning to introduce ES6 into their codebase may not want to apply this rule if the cost of migrating from `var` to `let` is too costly.

Version

Version

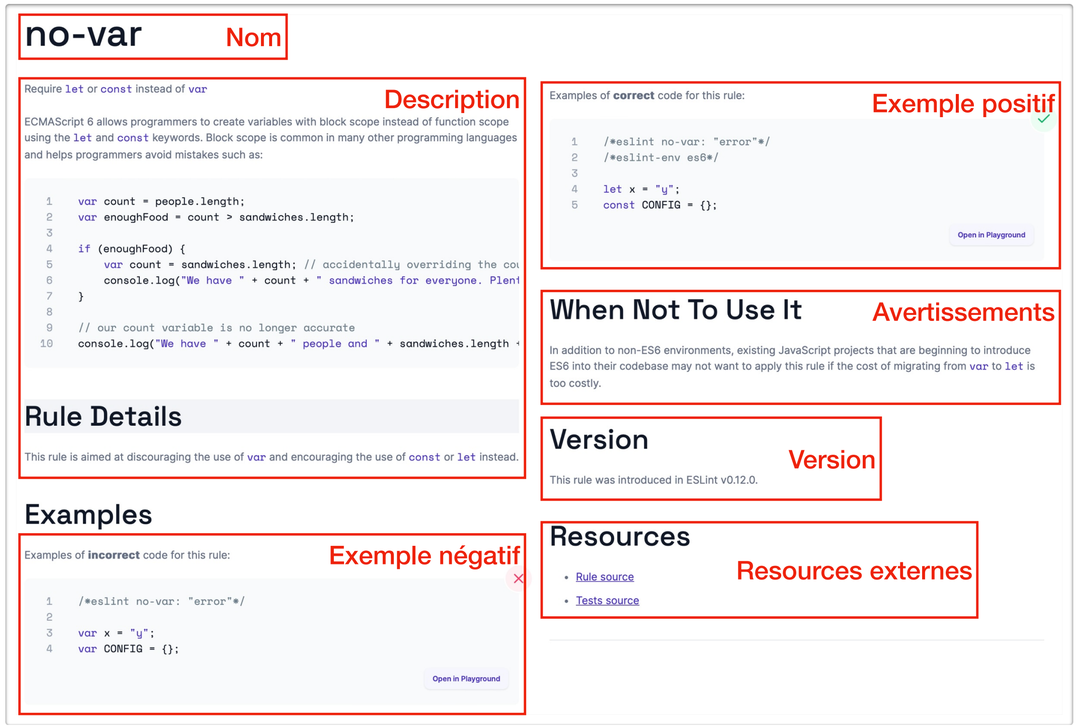
This rule was introduced in ESLint v0.12.0.

Resources

Resources externes

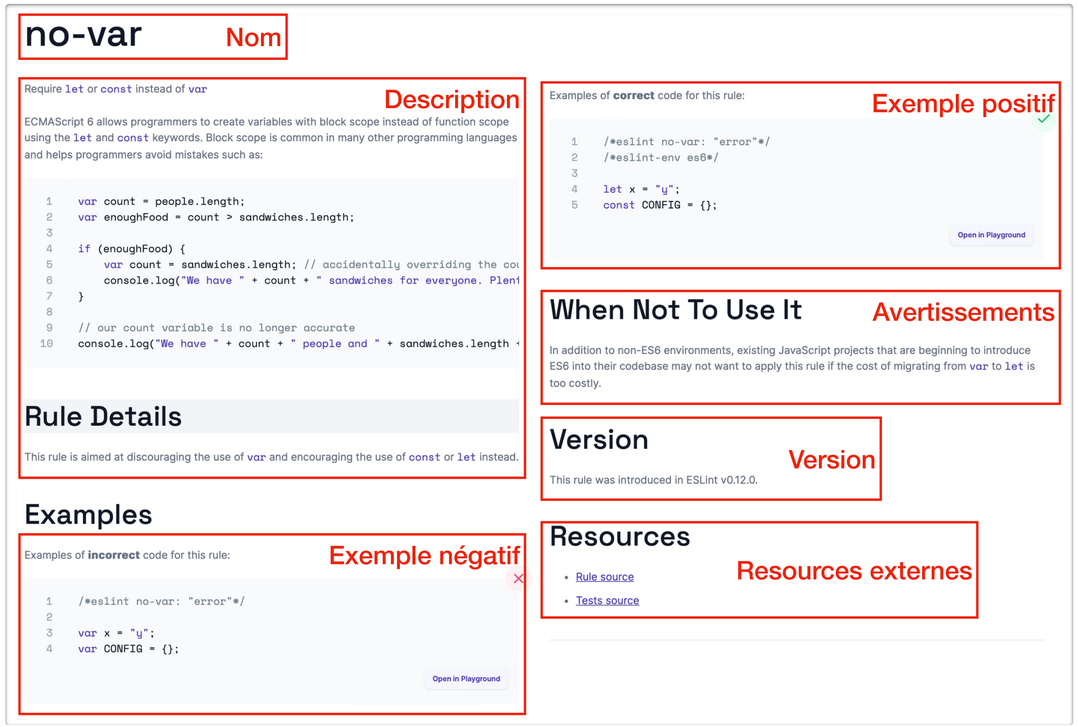
- Rule source
- Tests source

Analyse de la Documentation



Nom	Description	Exemple	Lien	
no-var	X	X		...
...				

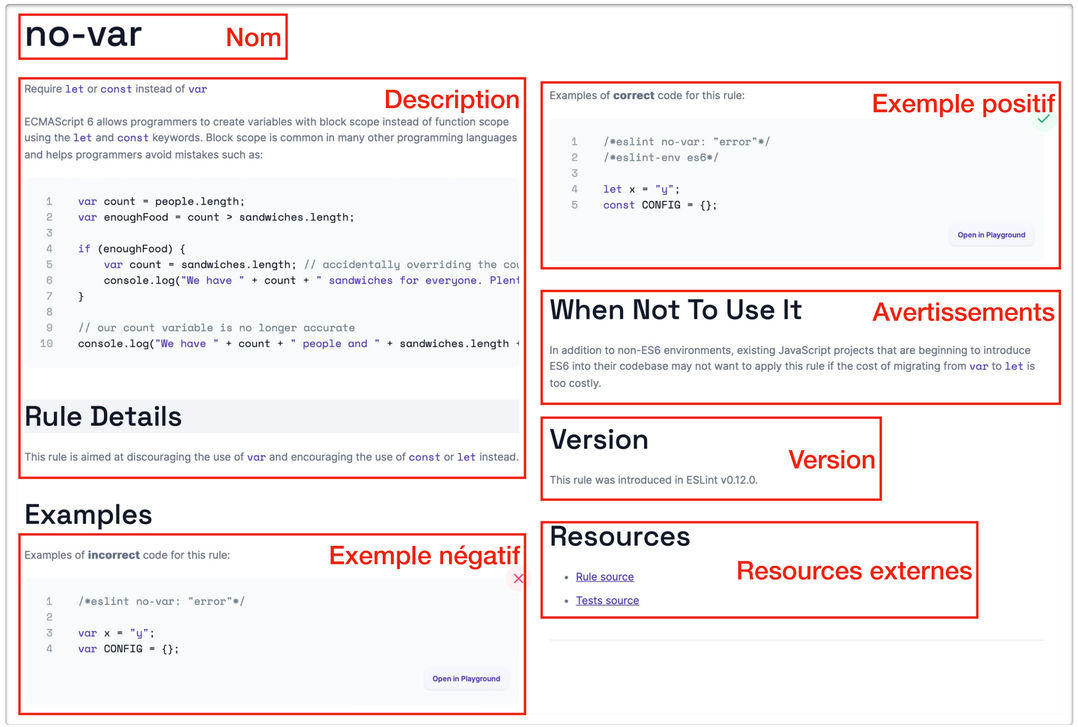
Analyse de la Documentation



Nom	Description	Exemple	Lien	
no-var	X	X		...
...				

7 langages

Analyse de la Documentation

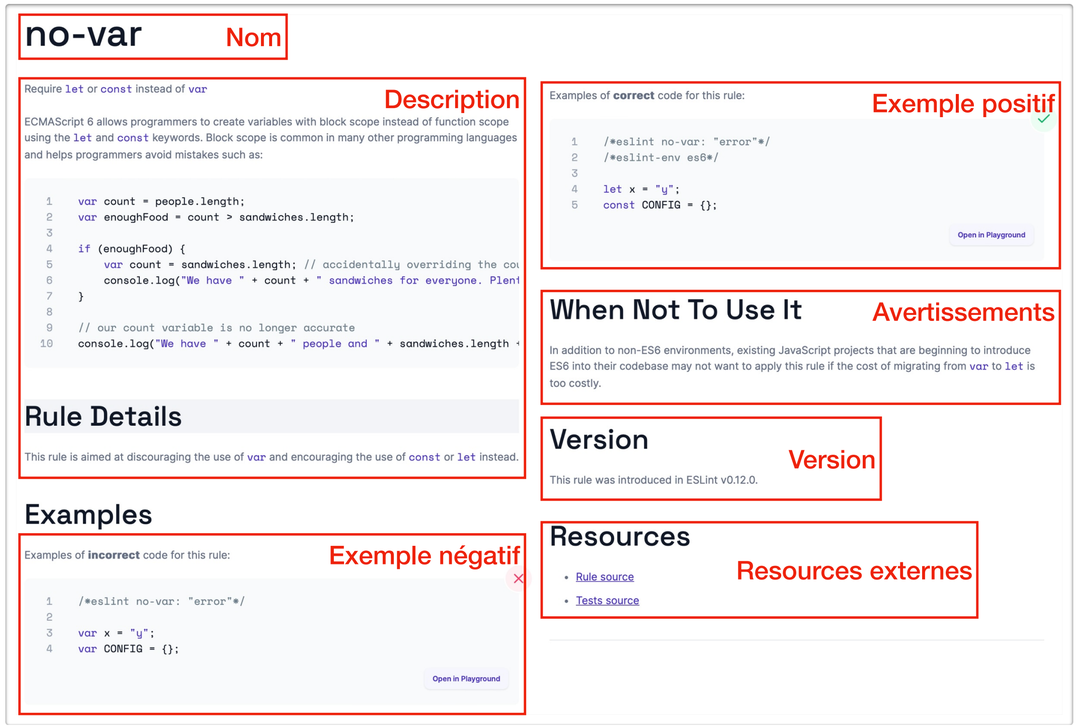


Nom	Description	Exemple	Lien	
no-var	X	X		...
...				

7 langages

14 outils

Analyse de la Documentation

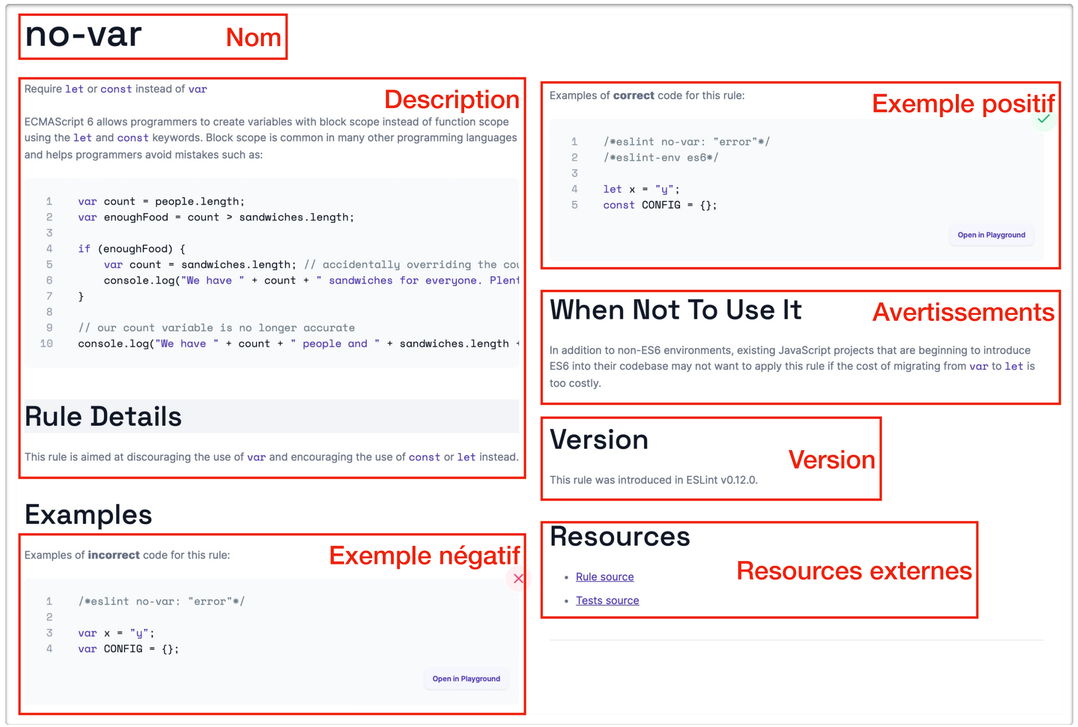


Nom	Description	Exemple	Lien	
no-var	X	X		...
...				

7 langages

14 outils + 2 multi-langages

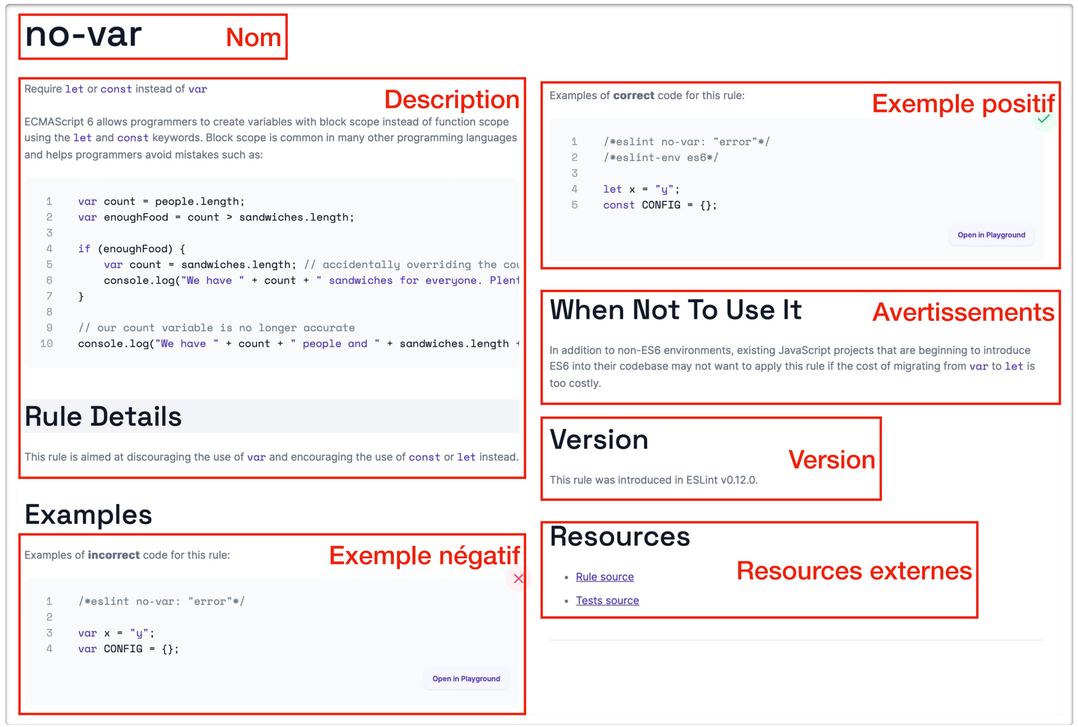
Analyse de la Documentation



Nom	Description	Exemple	Lien	
no-var	X	X		...
...				

7 langages | 14 outils + 2 multi-langages

Analyse de la Documentation

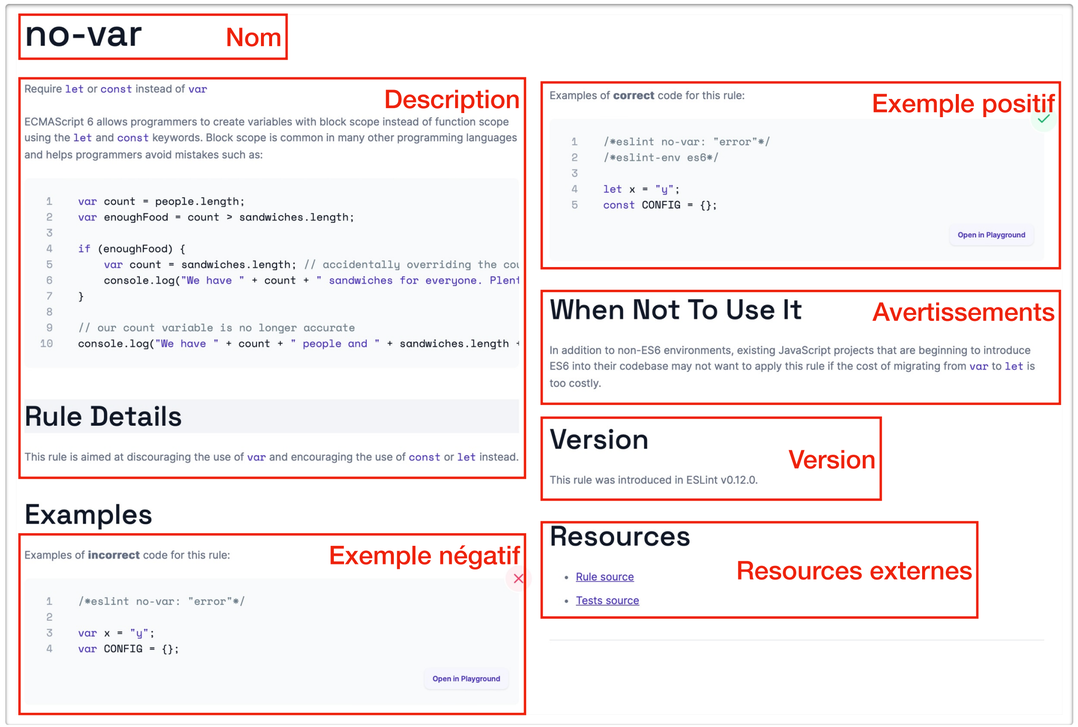


Nom	Description	Exemple	Lien	
no-var	X	X		...
...				

7 langages | 14 outils + 2 multi-langages

Des milliers de règles à analyser

Analyse de la Documentation

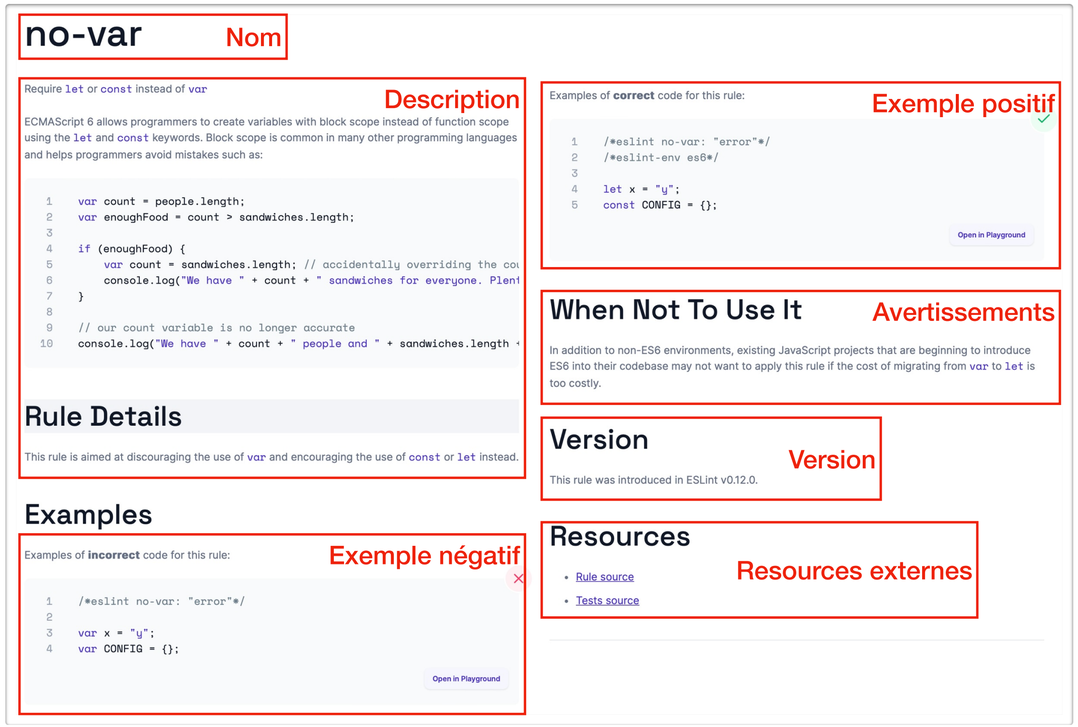


Nom	Description	Exemple	Lien	
no-var	X	X		...
...				

7 langages | 14 outils + 2 multi-langages

Des milliers de règles à analyser

Analyse de la Documentation

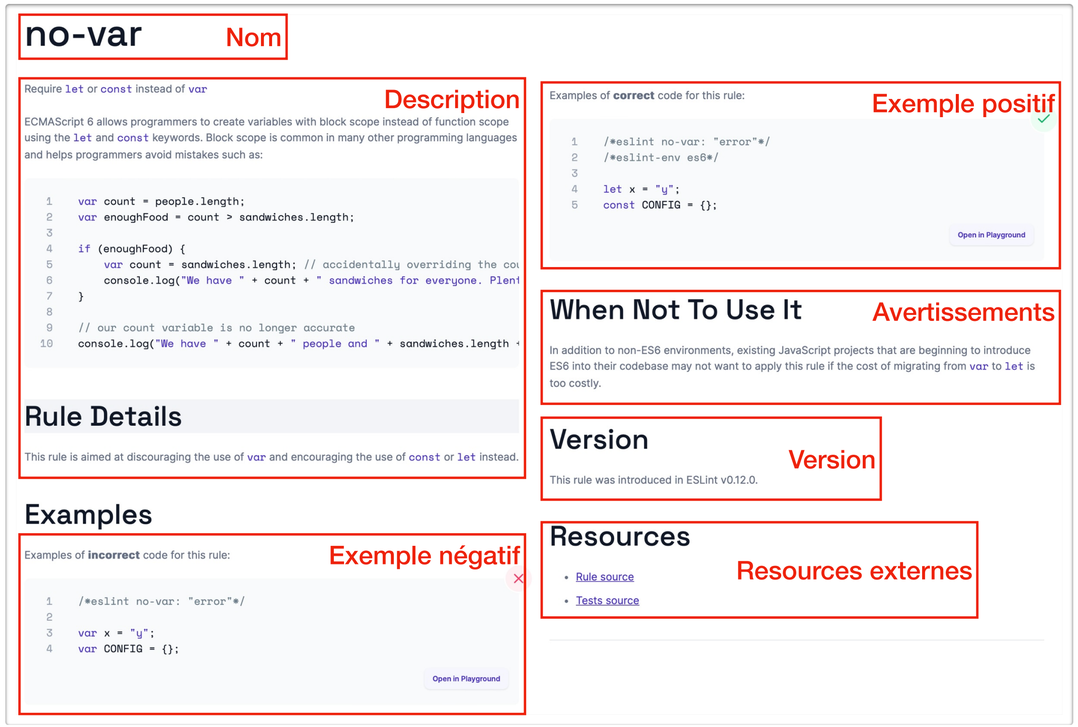


Nom	Description	Exemple	Lien	
no-var	X	X		...
...				

7 langages | 14 outils + 2 multi-langages

Processus de saturation

Analyse de la Documentation



Nom	Description	Exemple	Lien	
no-var	X	X		...
...				

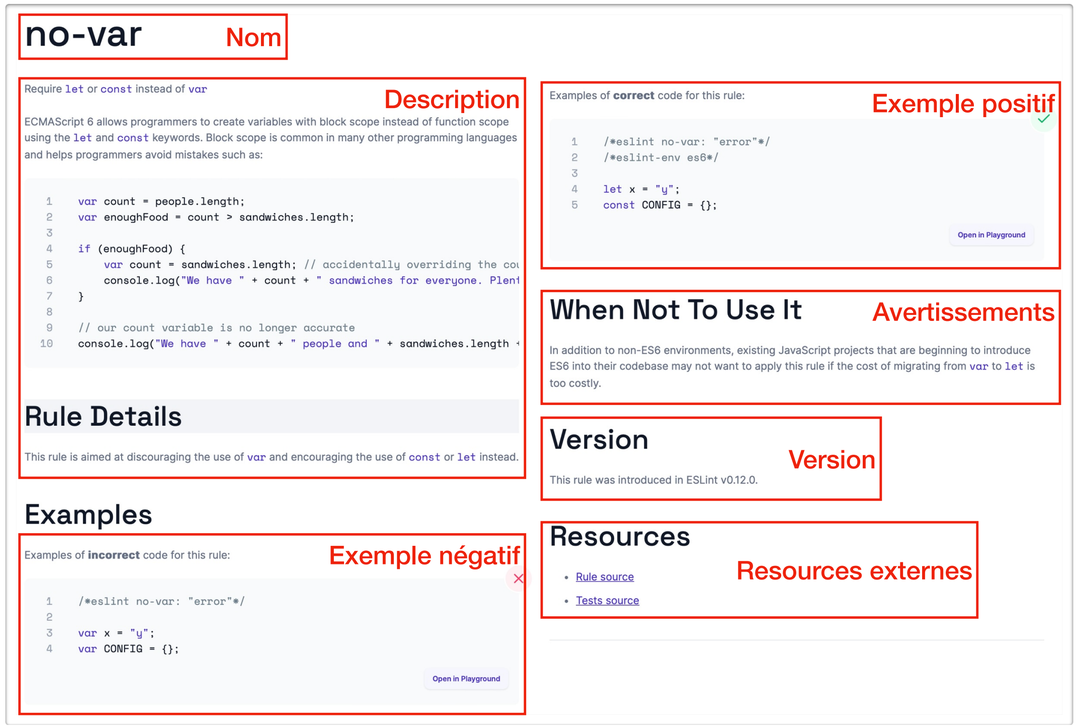
7 langages | 14 outils + 2 multi-langages

Processus de saturation

↓

5 règles consécutives sans nouveau concept

Analyse de la Documentation



Nom	Description	Exemple	Lien	
no-var	X	X		...
...				

7 langages | 14 outils + 2 multi-langages

119 règles analysées

État de la Documentation des Règles

Concepts	% de présence
Description	92
Code Example	87
Severity	34
Further Information	30
Since	29
Rule Definition	25
Configuration	22
Error Output	19
Auto Fix	15
Rule Set	11
Related Rules	8
When Not To Use It	6
Usage Example	5
Compatibility	3
IDE Fix	2

15 concepts

État de la Documentation des Règles

Thèmes	Concepts	% de présence
Comprehension	Description	92
	Code Example	87
	Further Information	30
	When Not To Use It	6
Usage	Since	29
	Configuration	22
	Error Output	19
	Auto Fix	15
	Usage Example	5
	Compatibility	3
	IDE Fix	2
Metadata	Severity	34
	Rule Definition	25
	Rule Set	11
	Related Rules	8

15 concepts 3 thèmes

État de la Documentation des Règles

Thèmes	Concepts	% de présence
Comprehension	Description	92
	Code Example	87
	Further Information	30
	When Not To Use It	6
Usage	Since	29
	Configuration	22
	Error Output	19
	Auto Fix	15
	Usage Example	5
	Compatibility	3
	IDE Fix	2
Metadata	Severity	34
	Rule Definition	25
	Rule Set	11
	Related Rules	8

15 concepts 3 thèmes

🔭 **Comprehension**

État de la Documentation des Règles

Thèmes	Concepts	% de présence
Comprehension	Description	92
	Code Example	87
	Further Information	30
	When Not To Use It	6
Usage	Since	29
	Configuration	22
	Error Output	19
	Auto Fix	15
	Usage Example	5
	Compatibility	3
	IDE Fix	2
Metadata	Severity	34
	Rule Definition	25
	Rule Set	11
	Related Rules	8

15 concepts 3 thèmes

🔭 **Comprehension**

↓

🔨 Taxonomie

🔎 Attentes développeur

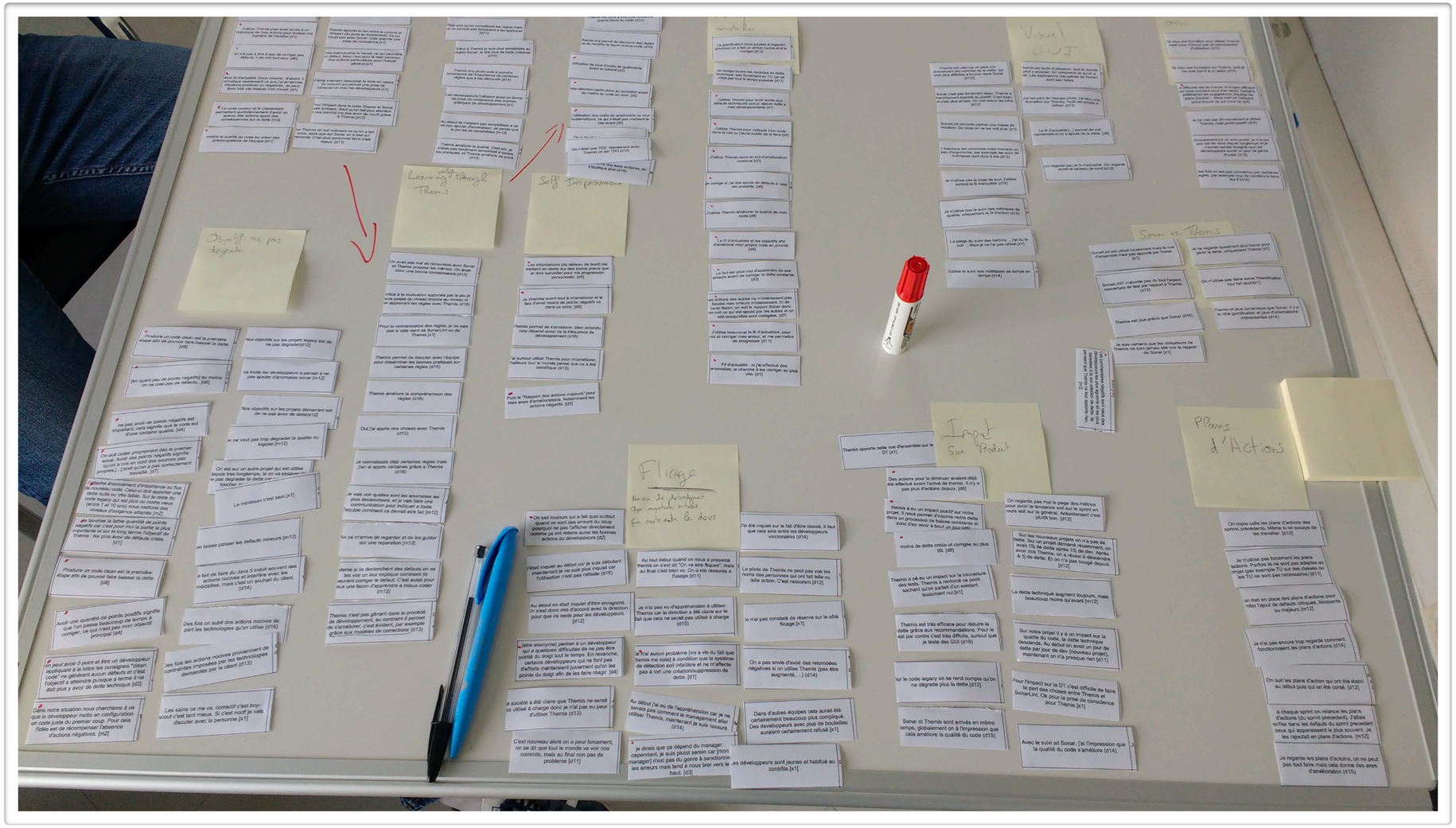
Taxonomie

Taxonomie

Open card-sorting

Taxonomie

Open card-sorting



Taxonomie

Open card-sorting

Taxonomie

Open card-sorting ⟶

Types de Contenu

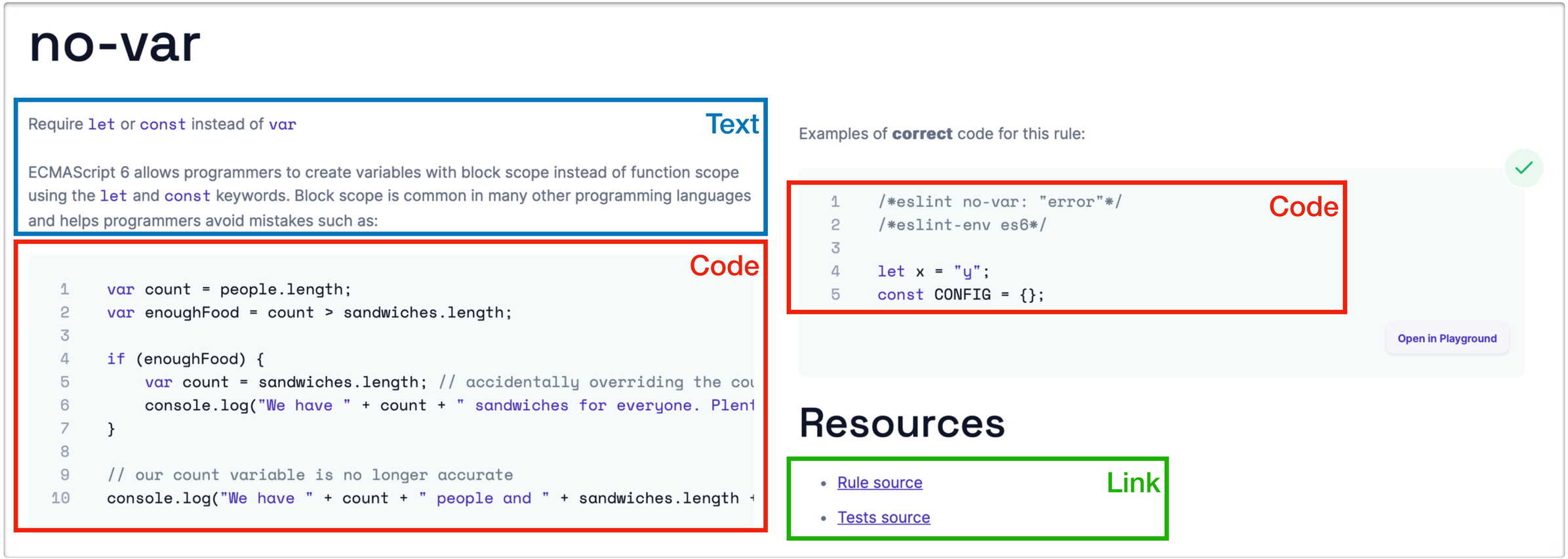
- Text
- Code
- Link

Taxonomie

Open card-sorting ⟶

Types de Contenu

- Text
- Code
- Link

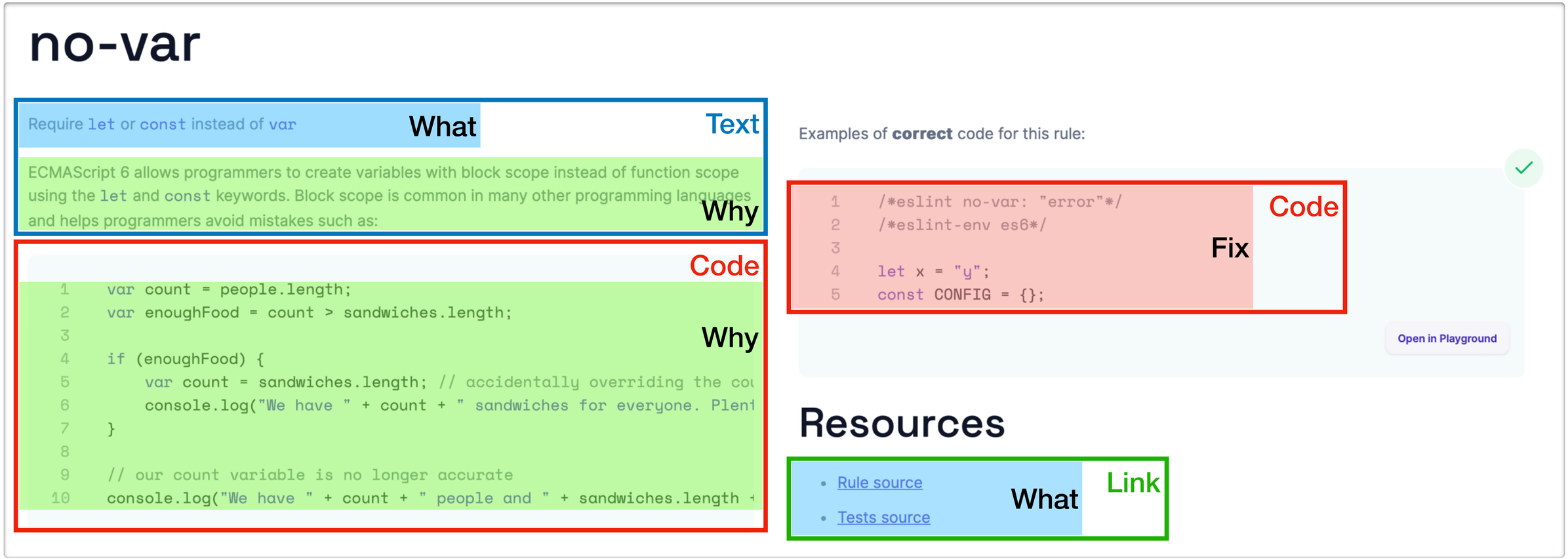


Taxonomie

Open card-sorting ⟶

Types de Contenu
- Text
- Code
- Link

Objectifs
- **What** (100%)
- **Why** (50%)
- **Fix** (77%)



Taxonomie

Open card-sorting ⟶

Types de Contenu
- Text
- Code
- Link

Objectifs
- **What** (100%)
- **Why** (50%)
- **Fix** (77%)

no-var

Require `let` or `const` instead of `var` — What — Text

ECMAScript 6 allows programmers to create variables with block scope instead of function scope using the `let` and `const` keywords. Block scope is common in many other programming languages and helps programmers avoid mistakes such as: — Why

Code

```
1   var count = people.length;
2   var enoughFood = count > sandwiches.length;
3
4   if (enoughFood) {
5       var count = sandwiches.length; // accidentally overriding the cou
6       console.log("We have " + count + " sandwiches for everyone. Plent
7   }
8
9   // our count variable is no longer accurate
10  console.log("We have " + count + " people and " + sandwiches.length +
```

Why

Examples of **correct** code for this rule:

Code

```
1   /*eslint no-var: "error"*/
2   /*eslint-env es6*/
3
4   let x = "y";
5   const CONFIG = {};
```

Fix

Open in Playground

Resources

- Rule source
- Tests source

What — Link

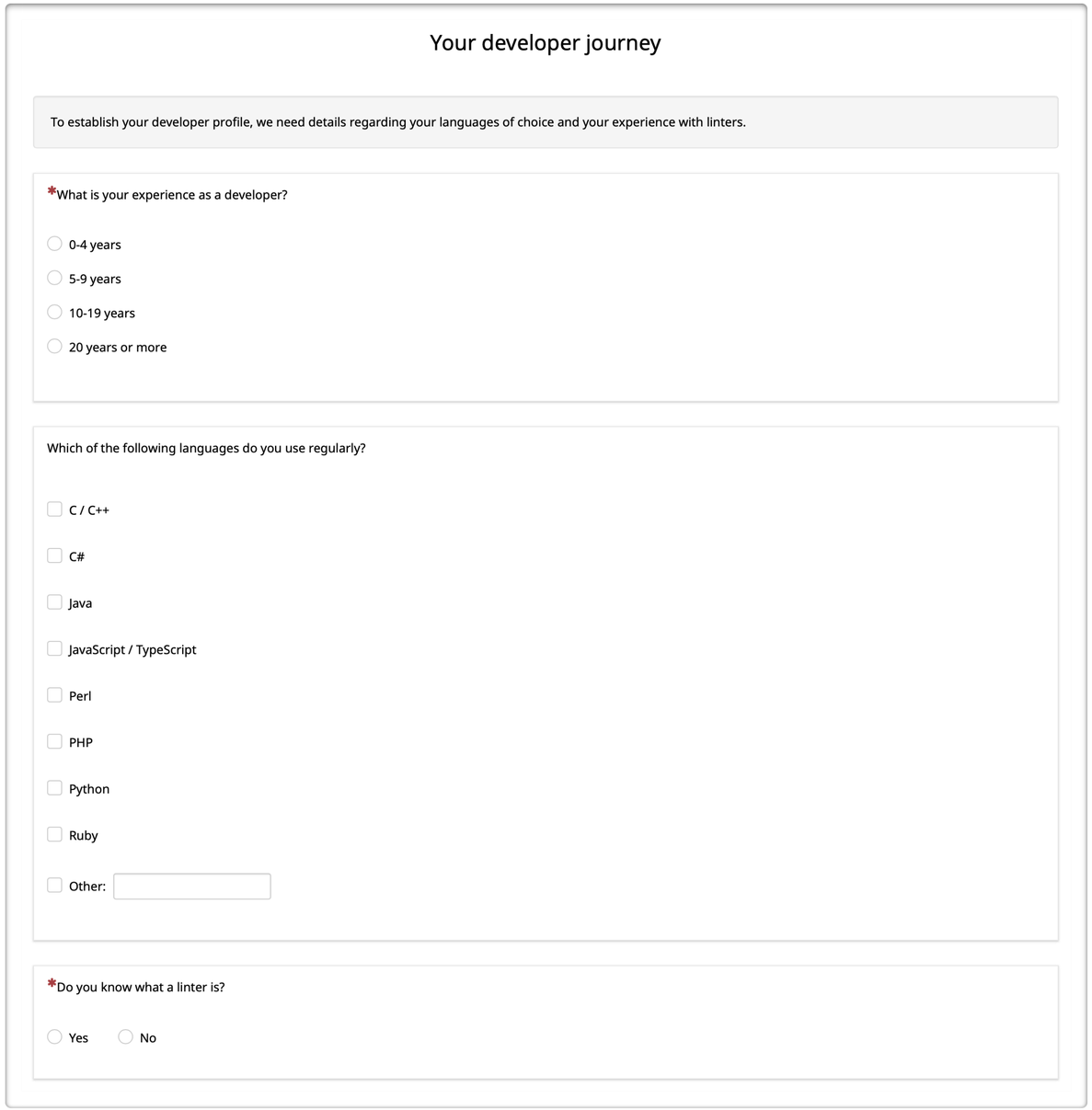
Enquête

4 étapes :

Enquête

4 étapes :

- Profil du développeur



Your developer journey

To establish your developer profile, we need details regarding your languages of choice and your experience with linters.

*What is your experience as a developer?

- 0-4 years
- 5-9 years
- 10-19 years
- 20 years or more

Which of the following languages do you use regularly?

- C / C++
- C#
- Java
- JavaScript / TypeScript
- Perl
- PHP
- Python
- Ruby
- Other:

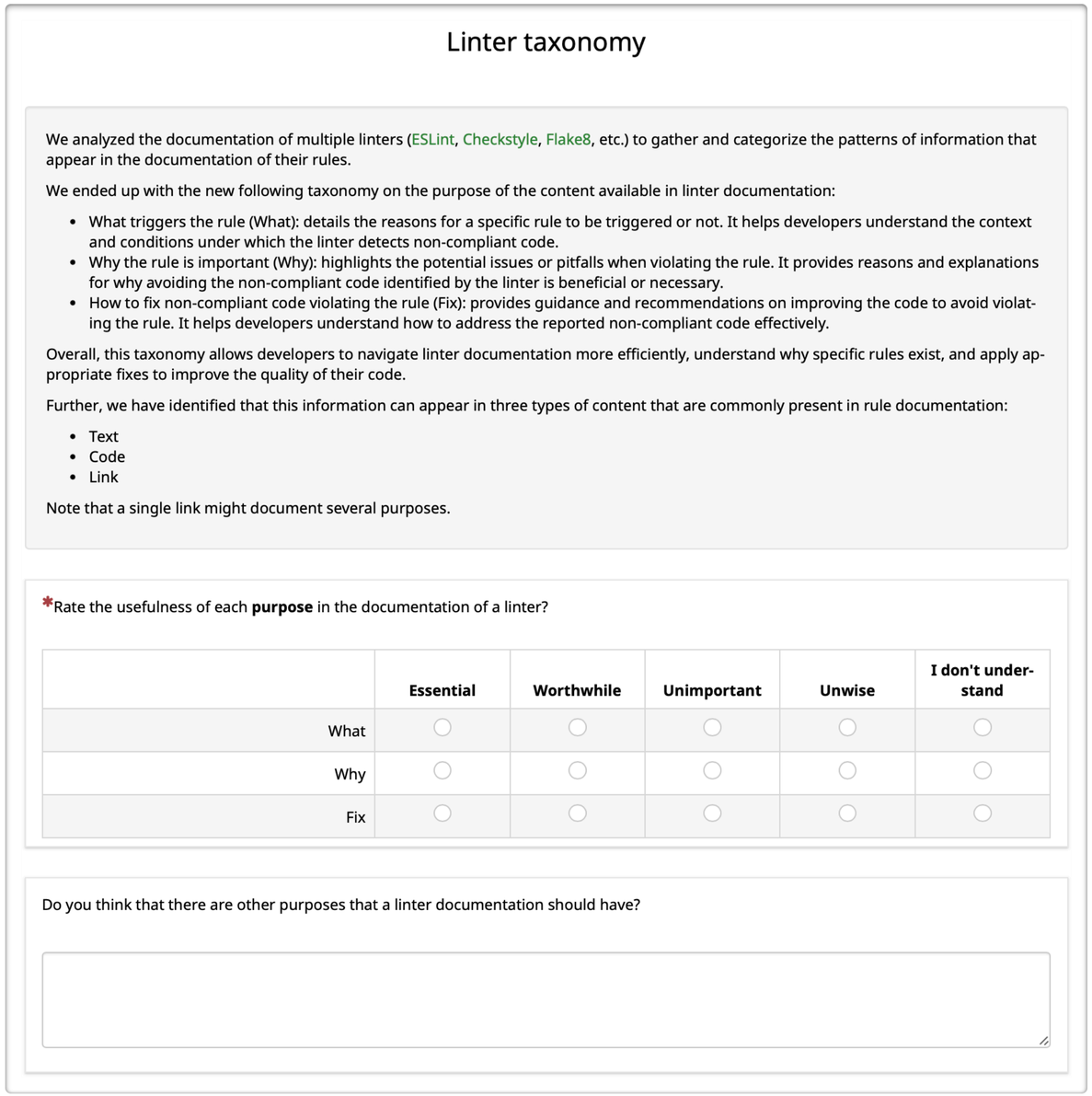
*Do you know what a linter is?

- Yes
- No

Enquête

4 étapes :

- Profil du développeur
- Évaluation de la Taxonomie



Linter taxonomy

We analyzed the documentation of multiple linters (ESLint, Checkstyle, Flake8, etc.) to gather and categorize the patterns of information that appear in the documentation of their rules.

We ended up with the new following taxonomy on the purpose of the content available in linter documentation:

- What triggers the rule (What): details the reasons for a specific rule to be triggered or not. It helps developers understand the context and conditions under which the linter detects non-compliant code.
- Why the rule is important (Why): highlights the potential issues or pitfalls when violating the rule. It provides reasons and explanations for why avoiding the non-compliant code identified by the linter is beneficial or necessary.
- How to fix non-compliant code violating the rule (Fix): provides guidance and recommendations on improving the code to avoid violating the rule. It helps developers understand how to address the reported non-compliant code effectively.

Overall, this taxonomy allows developers to navigate linter documentation more efficiently, understand why specific rules exist, and apply appropriate fixes to improve the quality of their code.

Further, we have identified that this information can appear in three types of content that are commonly present in rule documentation:

- Text
- Code
- Link

Note that a single link might document several purposes.

*Rate the usefulness of each **purpose** in the documentation of a linter?

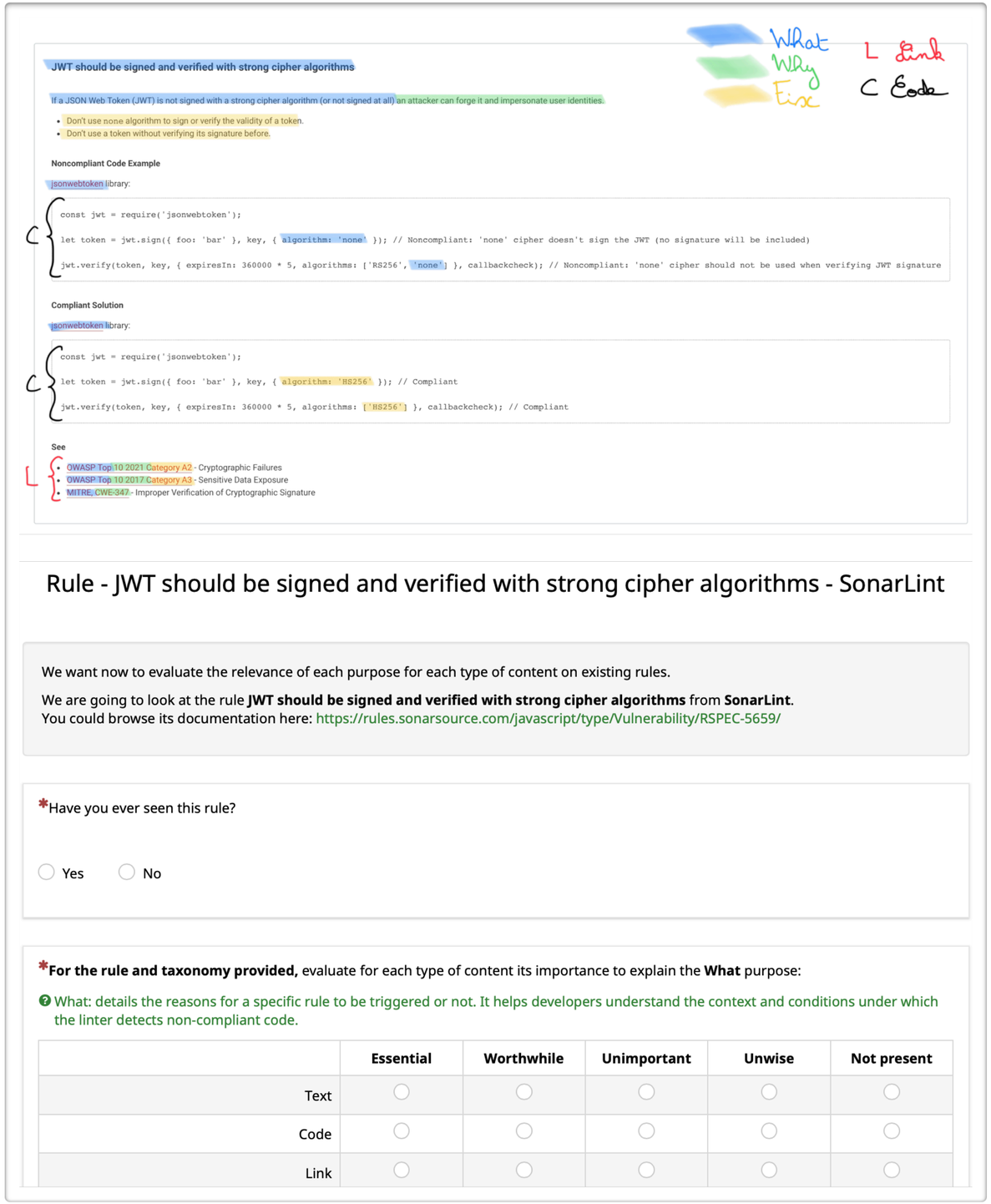
	Essential	Worthwhile	Unimportant	Unwise	I don't understand
What	○	○	○	○	○
Why	○	○	○	○	○
Fix	○	○	○	○	○

Do you think that there are other purposes that a linter documentation should have?

Enquête

4 étapes :

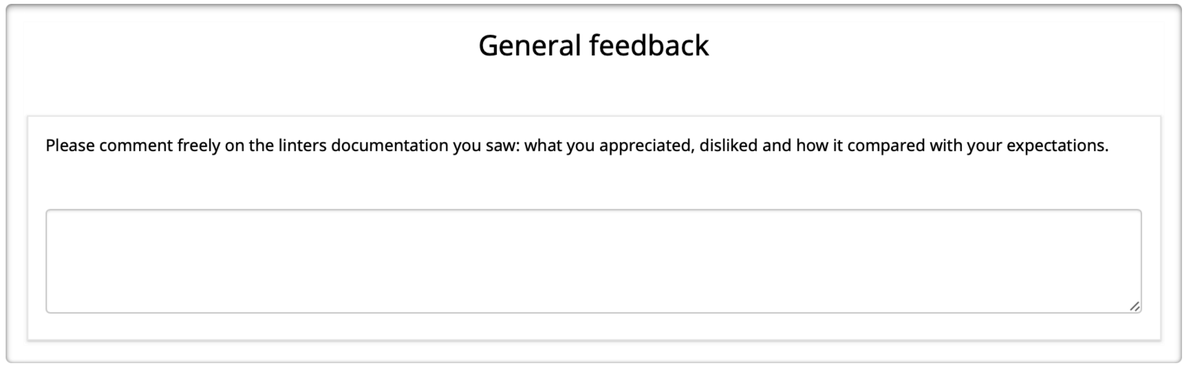
- Profil du développeur
- Évaluation de la Taxonomie
- Analyse de règles



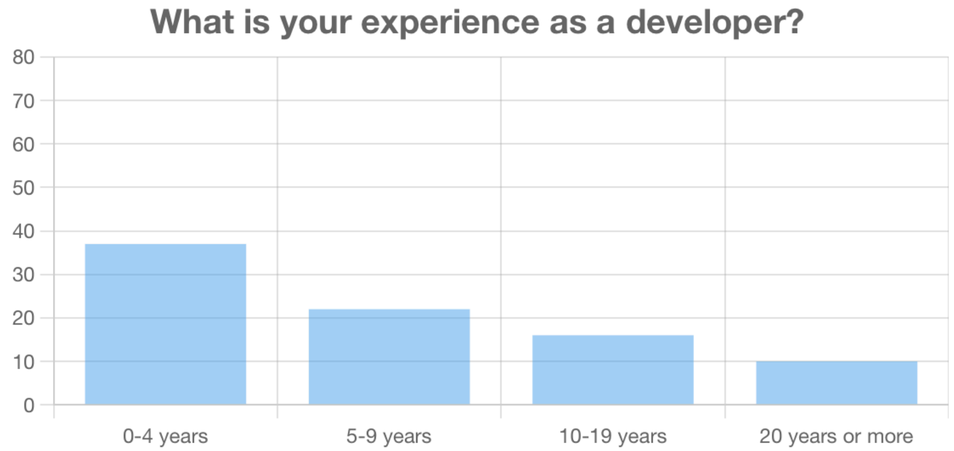
Enquête

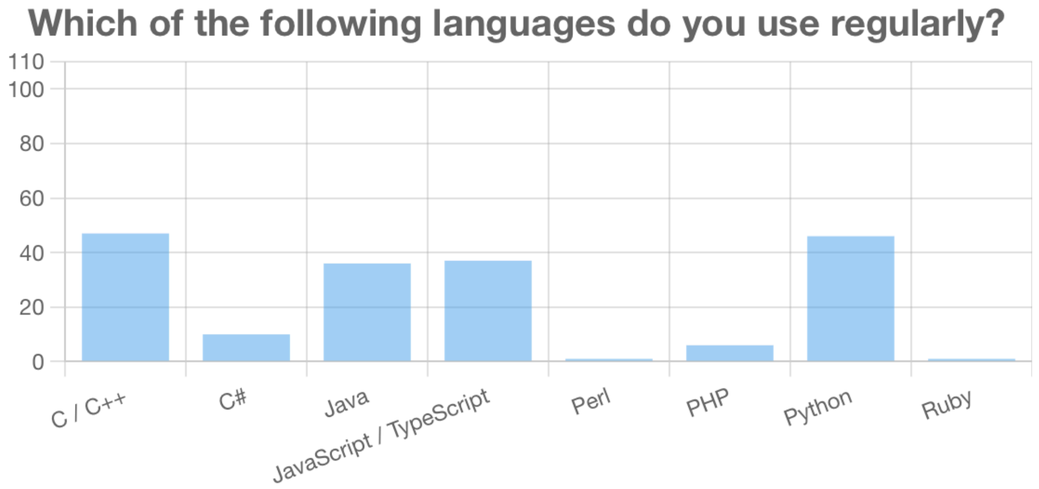
4 étapes :

- Profil du développeur
- Évaluation de la Taxonomie
- Analyse de règles
- Retours globaux

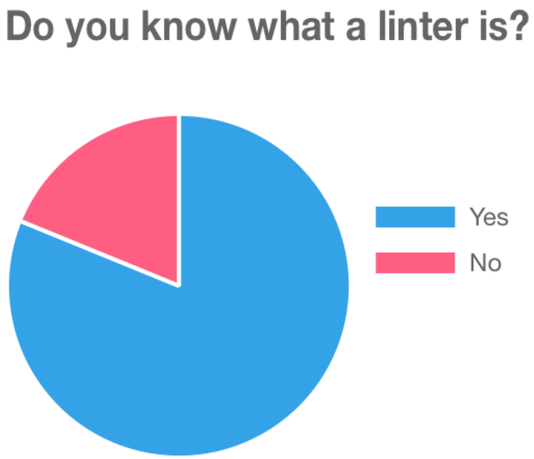


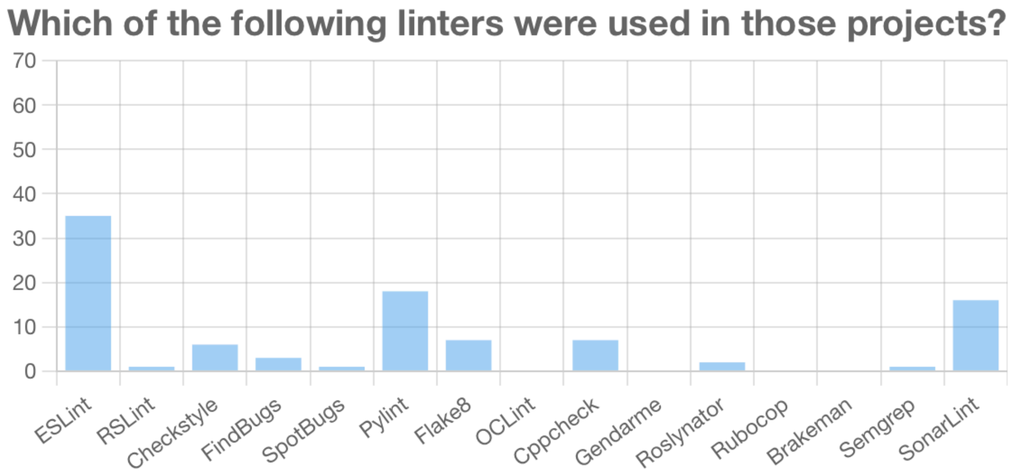
Profil des Participants





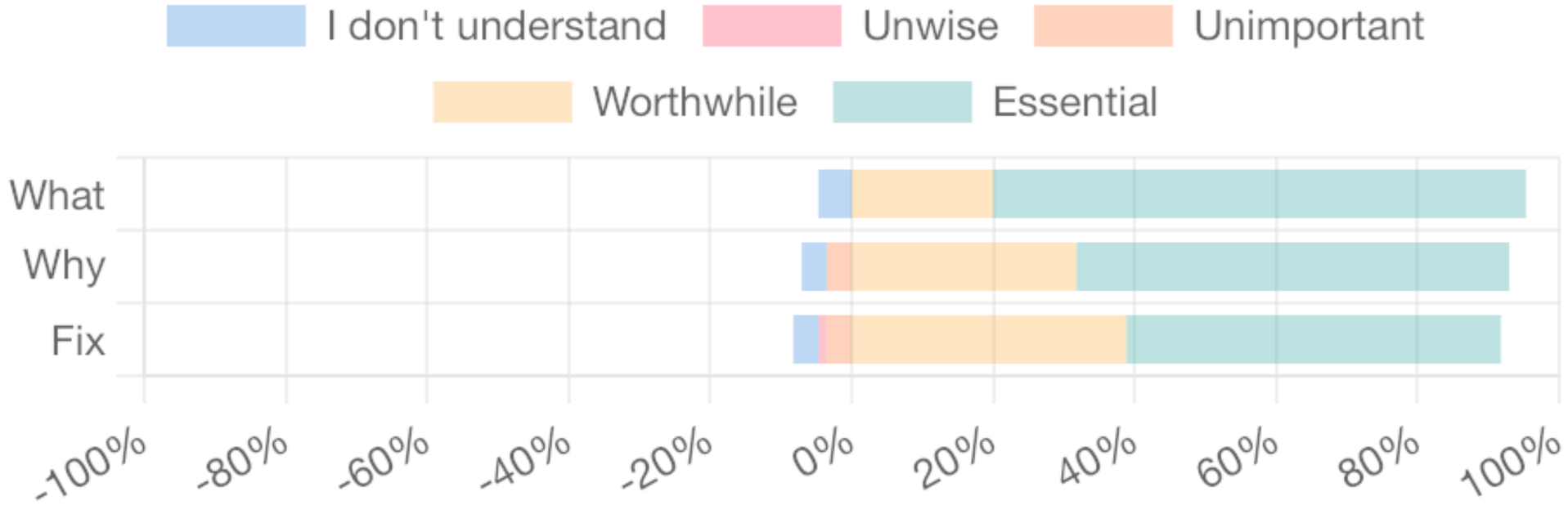
85 Participants





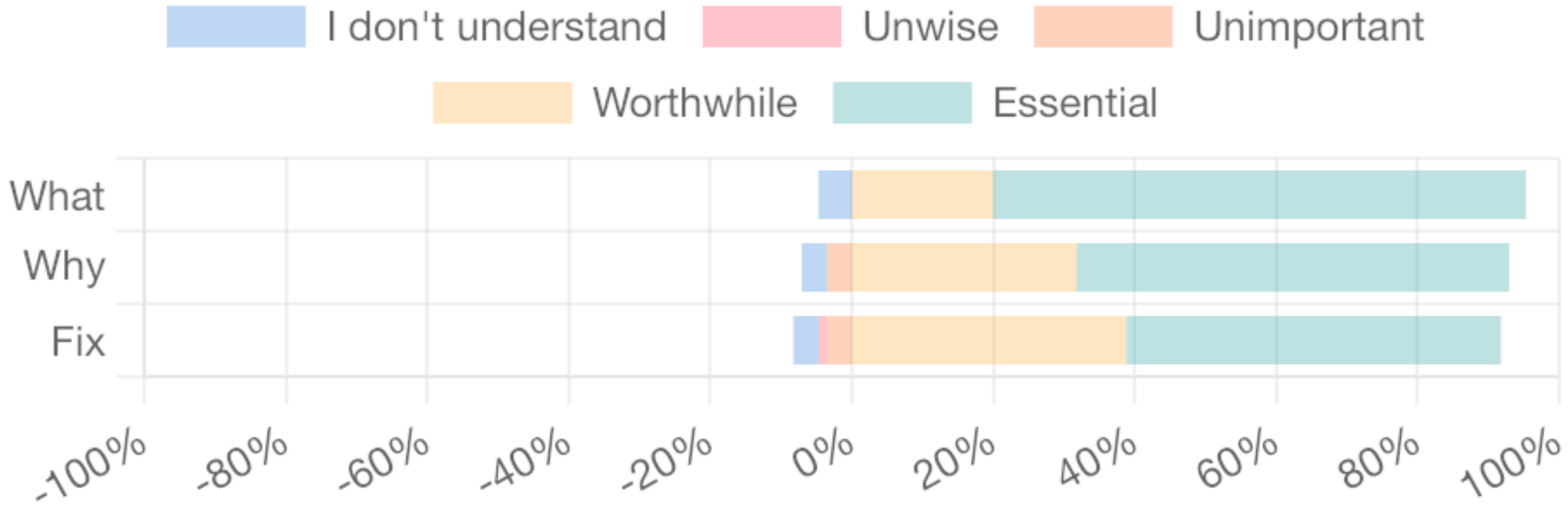
Résultats Quantitatifs

Importance de chaque **Objectif** dans la documentation



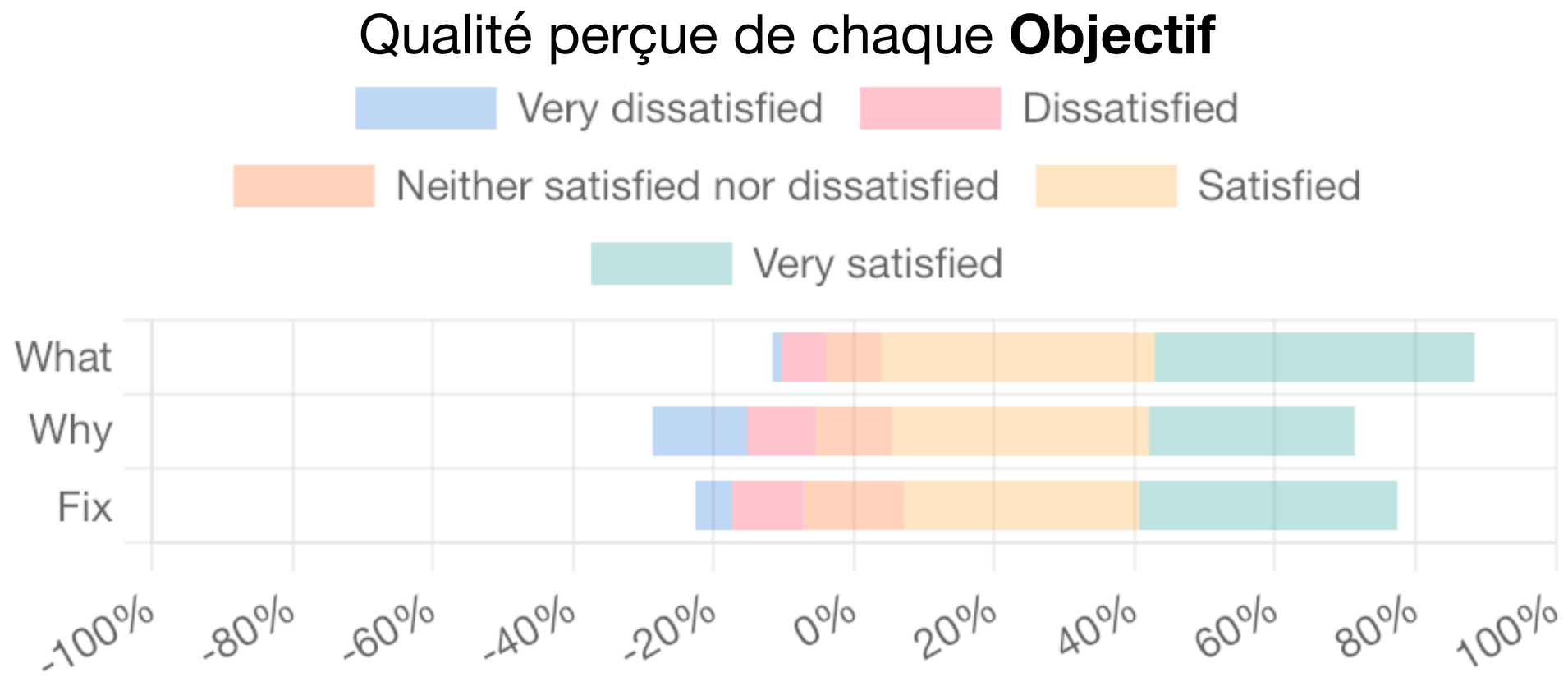
Résultats Quantitatifs

Importance de chaque **Objectif** dans la documentation

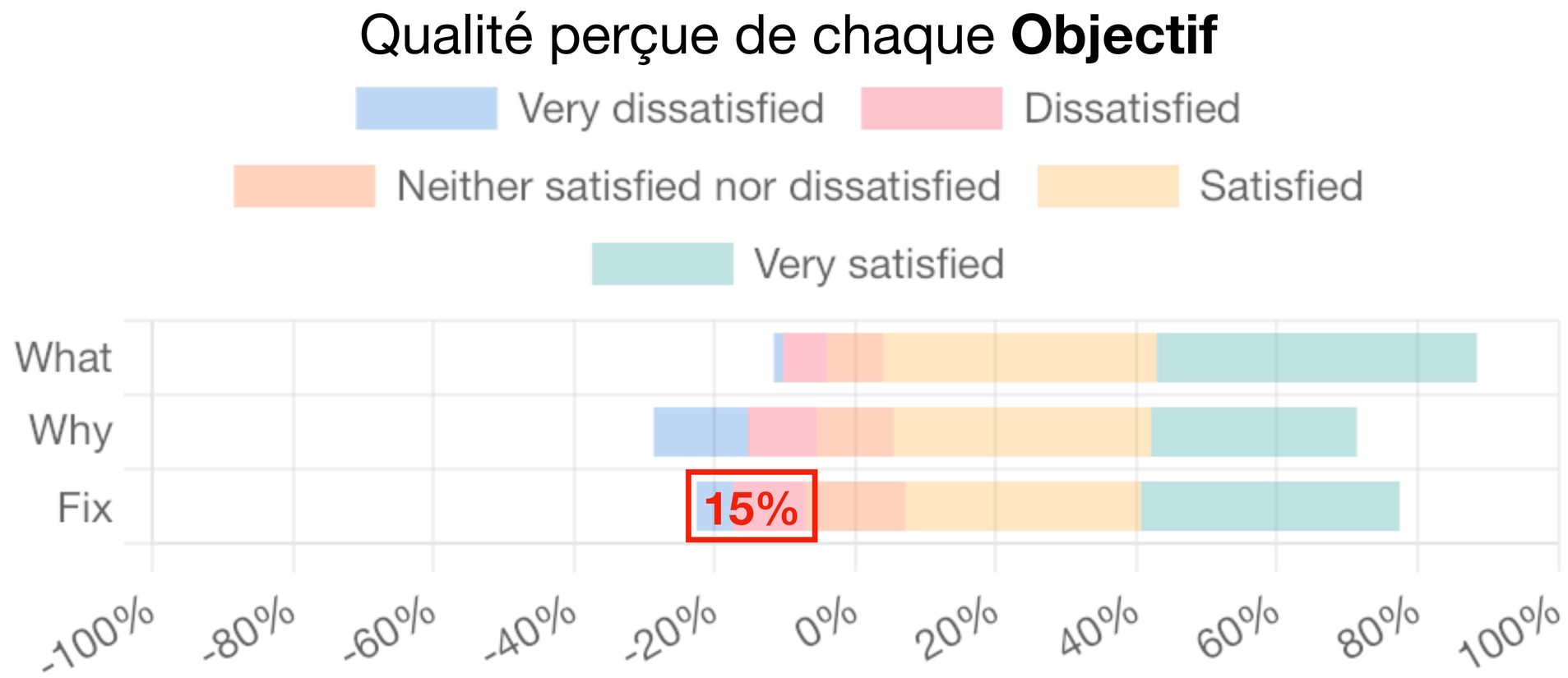


Chaque **Objectif** doit être documenté

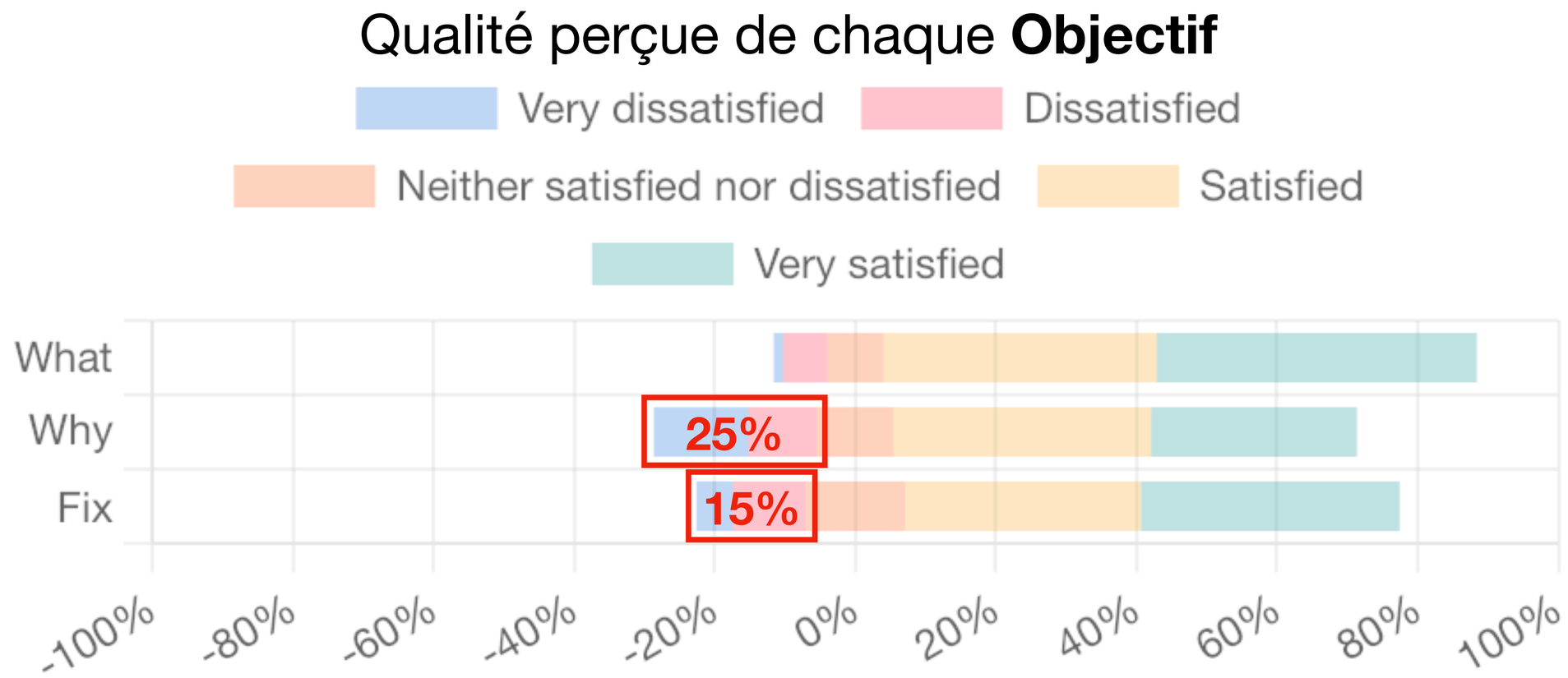
Résultats Quantitatifs



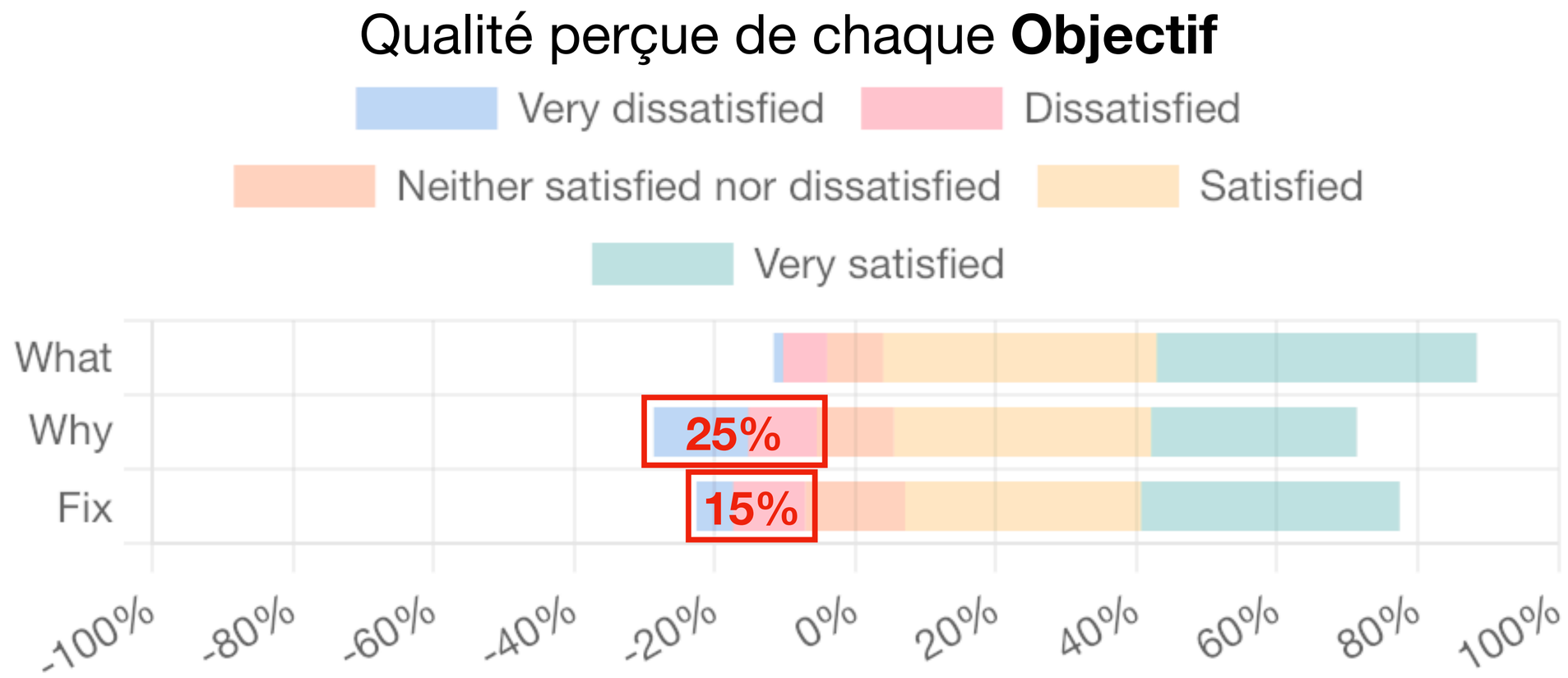
Résultats Quantitatifs



Résultats Quantitatifs



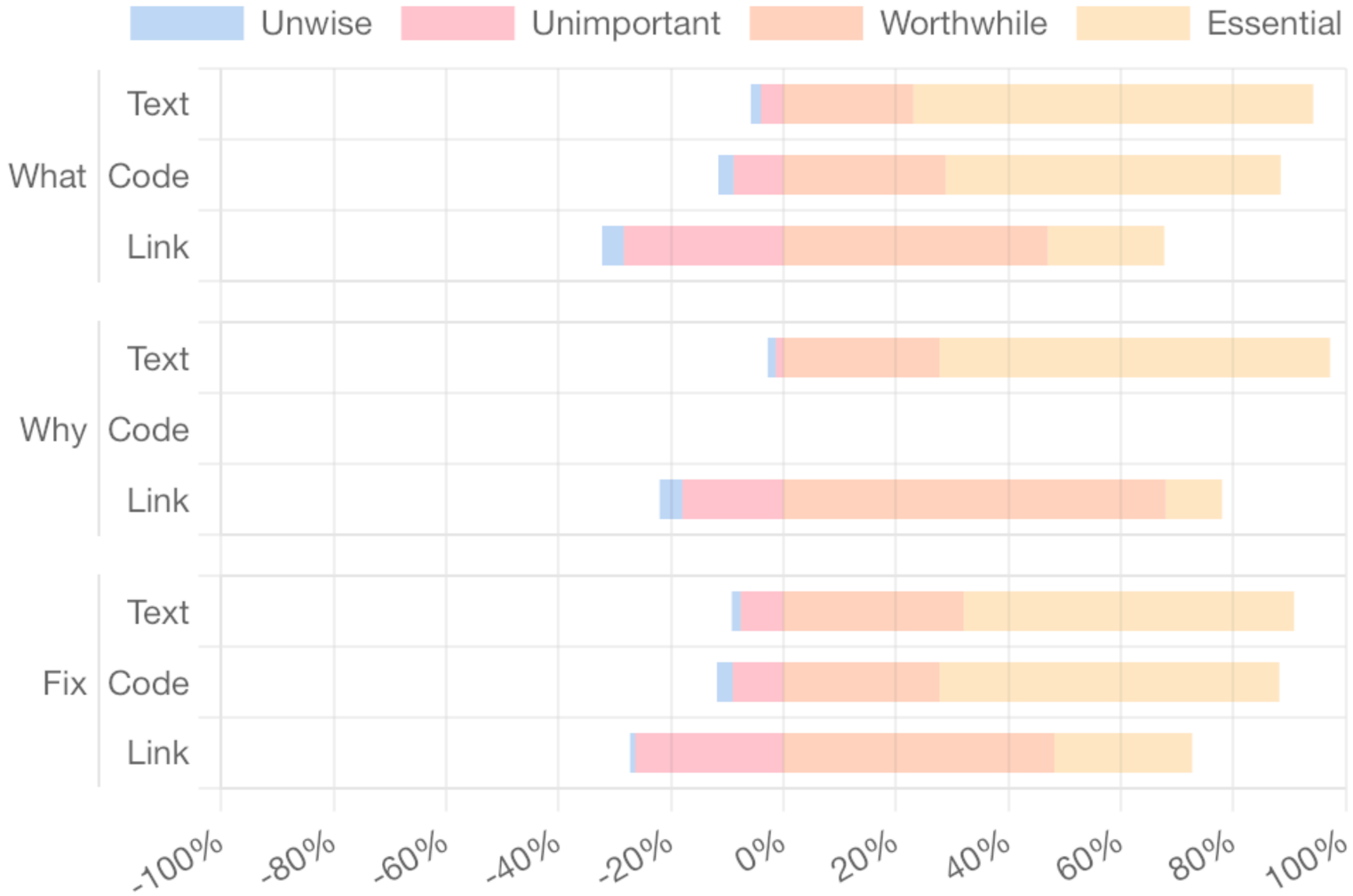
Résultats Quantitatifs



Why doit être amélioré

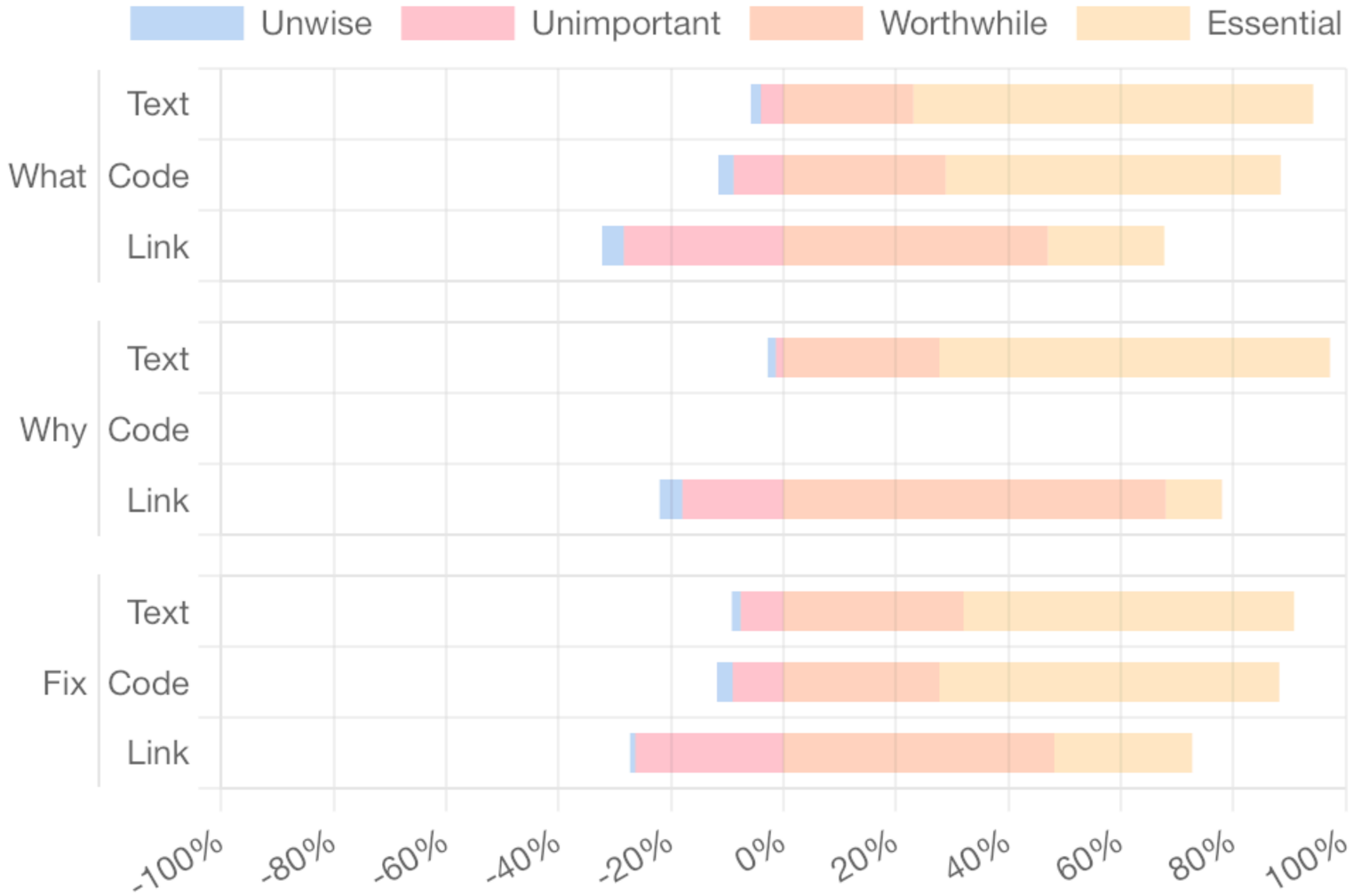
Résultats Quantitatifs

Importance des types de contenu pour documenter les **Objectifs**



Résultats Quantitatifs

Importance des types de contenu pour documenter les **Objectifs**

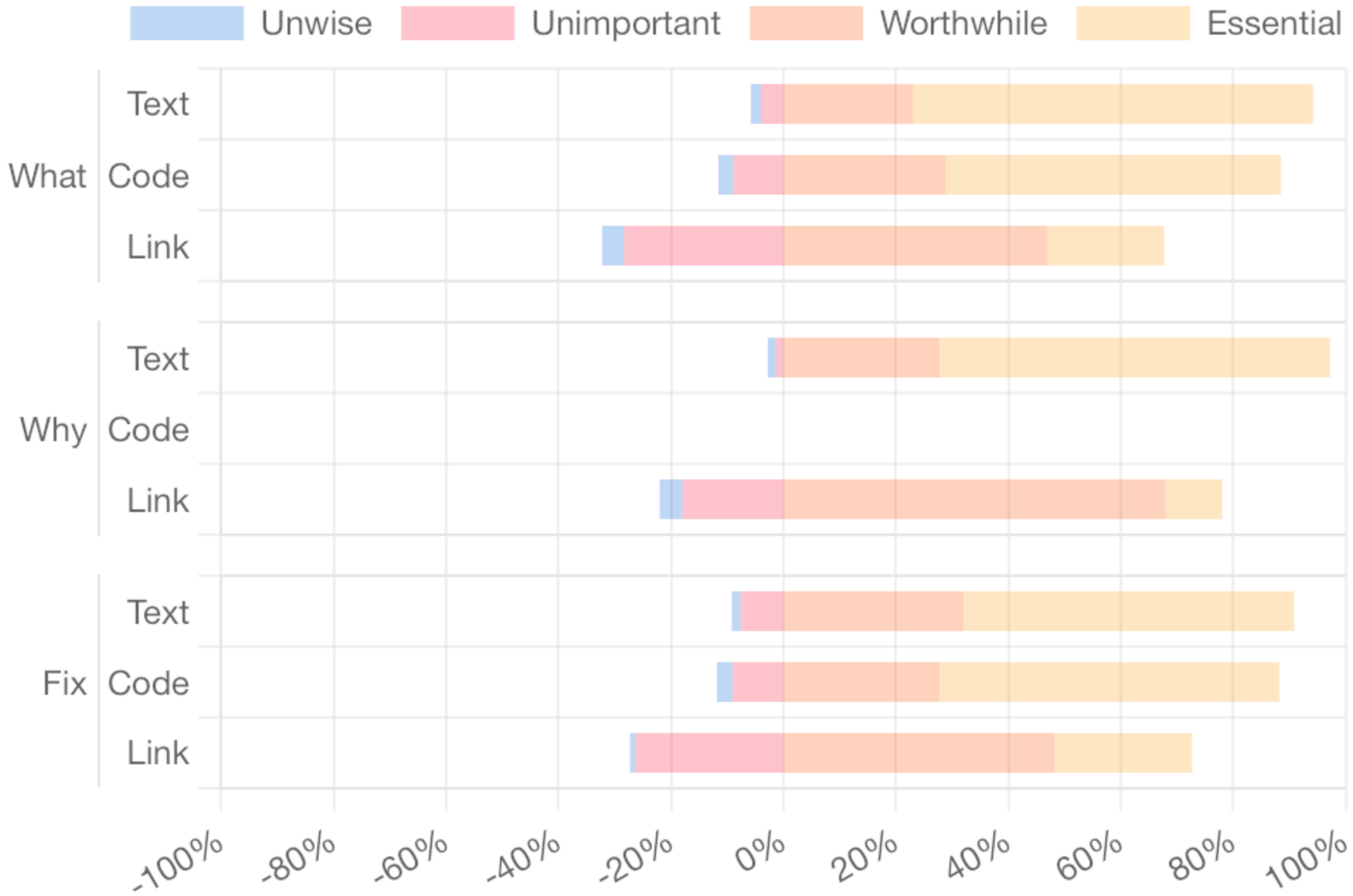


Pour documenter

- **What**: Text & Code
- **Why**: Text
- **Fix**: Text & Code

Résultats Quantitatifs

Importance des types de contenu pour documenter les **Objectifs**



Pour documenter

- **What**: Text & Code
- **Why**: Text
- **Fix**: Text & Code

Link avec parcimonie

Résultats Qualitatifs

Résultats Qualitatifs

Analyse
Thématique

Résultats Qualitatifs

Analyse Thématique

	A	B	C
1	**P**	**XP**	**For the Fix purpose, why do you think it is important or not important to be present in the documentation?**
2	P47	5-9 years	It makes it more concrete when the What is not clear enough
3	P111	10-19 years	The fix guidance can be helpful in many situations, but I don't consider it as essential: in most cases previous information should be enough to infer how to fix. Now in few cases it might be helpful to have examples of fixes.
4	P129	5-9 years	Some rules may be complicated so examples can help to understand what we need to do
5	P137	10-19 years	In my experience, the "fix" part is not that important, the linters rule are usually clear enough. But I believe examples are important for the programmers to get a better understanding of the rule. Also, some linters nowadays propose to automatically fix the problems if possible. The "fix" part then should explain what would be done automatically.
6	P68	20 years or more	At least not a cut/paste able correction, as many errors may happen, may be an example of correct (as in your example again)
7	P128	20 years or more	I don't want to Google for every linter warning, so it's useful to have a best practice recommendation provided directly by the linter.
8	P161	0-4 years	If the developer creates quality issue is not on purpose so after telling him the reason with the "why" it's important to show him compliant code to let him correct himself
9	P178	0-4 years	Fix is important because it shows exactly how to fix the mistake. However, it doesnt mean that the only way to fix is the way thats shown. So i dont think that its ass important
10	P181	0-4 years	is importat because show exectly how change
11	P189	0-4 years	Main examples of common errors in codes of a given language are important in documentation
12	P30	20 years or more	The fix is needed to illustrate how to avoid the problem explained in the why. It is likely the coder introduced a bad pattern/error due to lack of expertise; as such, it would be wrong to assume he/she will know how to fix it.
13	P52	0-4 years	The fix purpose can probably be omitted for the simpler rules, the previous two purposes should be enough to come up with a fix.
14	P65	10-19 years	Probably this can be under what or why, but in the end when a lint triggers, one obviously needs to have enough information to understand how to fix it otherwise they'll need to ask somebody else how to fix it which will cost a ton of time...
15	P104	5-9 years	Same uncertainty as above for the presence in the documentation. As to why it is important, I would say it is more important for new developers, who could learn good practice from such information, than experienced ones who probably already know it. But documentation being for everybody, it is sufficient to consider it important.
16	P116	10-19 years	It's another way to understand the rule, and it makes the fix easier to do.
17	P119	5-9 years	Related to the Why, to show why it is better. Sometimes the fix is non-trivial.
18	P133	10-19 years	I believe the Fix purpose is important, although not as much as the What and Why purpose. It is important because it helps developers fix a mistake or improve the code, but not as important, as a linting rule may already be helpful by detecting a situation requiring the developer's attention and explaining what is or may be wrong with it, even though it may not suggest how to fix it (the developer may still know how to fix it, or search for a fix).
19	P164	20 years or more	L'implémentation du "fix" ou comment on respecte la règle et de moi point de vue accessoire. Dans exemples de corrections sont nombreux sur Internet. Ce qui est important au fond, c'est la règle et sa finalité...
20	P235	20 years or more	When the fix is rather straightforward, it is useful to know it as it can help understanding the rule and if the fix can be applied automatically it allows to preview the changes. However, in other cases it can lead beginners to apply a cascade of bad decisions made to satisfy the linter at the expense of the design consistency.
21	P24	5-9 years	This help to easier fix the non-compliant code and to know a way to avoid this
22	P33	10-19 years	it's really saving time when fix is clearly suggested, it avoids also browsing all the web to read maaaaaaany discussions on the topic... ideal thing is to just click a button to 'autofix'

+ ≡ What Why Fix Other General

Résultats Qualitatifs

Analyse Thématique

	A	B	C	D	E	F	G	H	I	J	
1	**P**	**XP**	**For the Fix purpose, why do you think it is important or not important to be present in the documentation?**	**Fixes not needed / less important**	**Fixes are essential / important**	**Foster learning**	**Help making the warning go away**	**Useful when the fix is not obvious**	**Documented fix might not be the best one**	**Allow developers to save time (less googling or questions)**	**One-clic autofix**
2	P47	5-9 years	It makes it more concrete when the What is not clear enough								
3	P111	10-19 years	The fix guidance can be helpful in many situations, but I don't consider it as essential: in most cases previous information should be enough to infer how to fix. Now in few cases it might be helpful to have examples of fixes.	X							
4	P129	5-9 years	Some rules may be complicated so examples can help to understand what we need to do					X			
5	P137	10-19 years	In my experience, the "fix" part is not that important, the linters rule are usually clear enough. But I believe examples are important for the programmers to get a better understanding of the rule. Also, some linters nowadays propose to automatically fix the problems if possible. The "fix" part then should explain what would be done automatically.	X							X
6	P68	20 years or more	At least not a cut/paste able correction, as many errors may happen, may be an example of correct (as in your example again)								
7	P128	20 years or more	I don't want to Google for every linter warning, so it's useful to have a best practice recommendation provided directly by the linter.							X	
8	P161	0-4 years	If the developer creates quality issue is not on purpose so after telling him the reason with the "why" it's important to show him compliant code to let him correct himself								
9	P178	0-4 years	Fix is important because it shows exactly how to fix the mistake. However, it doesnt mean that the only way to fix is the way thats shown. So i dont think that its ass important		X				X		
10	P181	0-4 years	is importat because show exectly how change		X						
11	P189	0-4 years	Main examples of common errors in codes of a given language are important in documentation								
12	P30	20 years or more	The fix is needed to illustrate how to avoid the problem explained in the why. It is likely the coder introduced a bad pattern/error due to lack of expertise; as such, it would be wrong to assume he/she will know how to fix it.			X					
13	P52	0-4 years	The fix purpose can probably be omitted for the simpler rules, the previous two purposes should be enough to come up with a fix.	X							
14	P65	10-19 years	Probably this can be under what or why, but in the end when a lint triggers, one obviously needs to have enough information to understand how to fix it otherwise they'll need to ask somebody else how to fix it which will cost a ton of time...			X	X			X	
15	P104	5-9 years	Same uncertainty as above for the presence in the documentation. As to why it is important, I would say it is more important for new developers, who could learn good practice from such information, than experienced ones who probably already know it. But documentation being for everybody, it is sufficient to consider it important.	X		X					
16	P116	10-19 years	It's another way to understand the rule, and it makes the fix easier to do.							X	
17	P119	5-9 years	Related to the Why, to show why it is better. Sometimes the fix is non-trivial.					X			
18	P133	10-19 years	I believe the Fix purpose is important, although not as much as the What and Why purpose. It is important because it helps developers fix a mistake or improve the code, but not as important, as a linting rule may already be helpful by detecting a situation requiring the developer's attention and explaining what is or may be wrong with it, even though it may not suggest how to fix it (the developer may still know how to fix it, or search for a fix).	X							
19	P164	20 years or more	L'implémentation du "fix" ou comment on respecte la règle et de moi point de vue accessoire. Dans exemples de corrections sont nombreux sur Internet. Ce qui est important au fond, c'est la règle et sa finalité...	X							
20	P235	20 years or more	When the fix is rather straightforward, it is useful to know it as it can help understanding the rule and if the fix can be applied automatically it allows to preview the changes. However, in other cases it can lead beginners to apply a cascade of bad decisions made to satisfy the linter at the expense of the design consistency.						X		X
21	P24	5-9 years	This help to easier fix the non-compliant code and to know a way to avoid this			X	X				
22	P33	10-19 years	it's really saving time when fix is clearly suggested, it avoids also browsing all the web to read maaaaaaany discussions on the topic... ideal thing is to just click a button to 'autofix'								

What | Why | Fix | Other | General

Résultats Qualitatifs

Résultats Qualitatifs

Apprentissage ⚡ Gain de temps

Résultats Qualitatifs

Apprentissage ⚡ Gain de temps

Utilisation d’un template structuré, avec

Résultats Qualitatifs

Apprentissage ⚡ Gain de temps

Utilisation d’un template structuré, avec

- résumé

Résultats Qualitatifs

Apprentissage ⚡ Gain de temps

Utilisation d’un template structuré, avec

- résumé
- exemples de code

Résultats Qualitatifs

Apprentissage ⚡ Gain de temps

Utilisation d’un template structuré, avec

- résumé
- exemples de code

Utilisation des liens externes avec parcimonie

En résumé

Question de Recherche

Qu’est-ce qu’une bonne documentation pour une pratique de code ?

17

État de la Documentation des Règles

Thèmes	Concepts	% de présence
Comprehension	Description	92
	Code Example	87
	Further Information	30
	When Not To Use It	6
Usage	Since	29
	Configuration	22
	Error Output	19
	Auto Fix	15
	Usage Example	5
	Compatibility	3
	IDE Fix	2
Metadata	Severity	34
	Rule Definition	25
	Rule Set	11
	Related Rules	8

15 concepts 3 thèmes

🔭 **Comprehension**

↓

🔨 Taxonomie

🔎 Attentes développeur

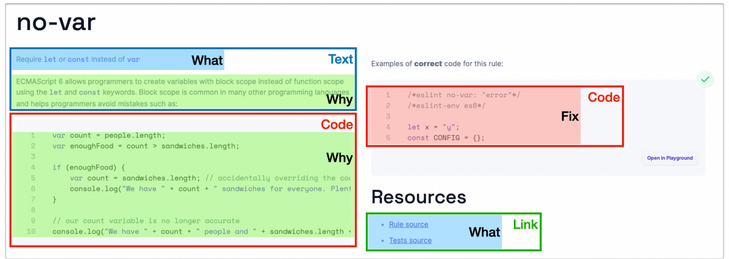
20

Taxonomie

Open card-sorting ⟶

Types de Contenu
- Text
- Code
- Link

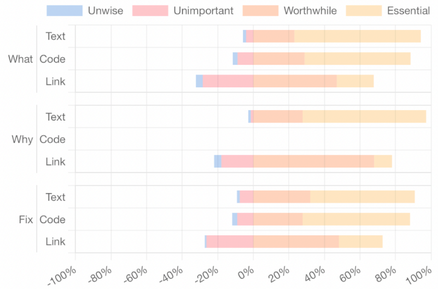
Objectifs
- **What** (100%)
- **Why** (50%)
- **Fix** (77%)



21

Résultats Quantitatifs

Importance des types de contenu pour documenter les **Objectifs**



Pour documenter
- **What**: Text & Code
- **Why**: Text
- **Fix**: Text & Code

Link avec parsimonie

26

Résultats Qualitatifs

Apprentissage ⚡ Gain de temps

Utilisation d’un template structuré, avec
- résumé
- exemples de code

Utilisation des liens externes avec parcimonie

27

Impacts pour [⋋] packmind

Impacts pour packmind

Interface d’affichage d’une pratique

Impacts pour [⋋] packmind

Interface d’affichage d’une pratique

- Mettre en avant la correction

Add a timestamp as query on request to avoid cache issues

Created by Corentin

Cache

Documentation | Detection

Multi-file example | Other examples

getCompleteResults.ts

No description

```
65    codings: { [key: string]: Codings };
66  };
67
68  const fetchAllData = async (): Promise<AllData> => {
      Add a timestamp as query on request to avoid cache issues
69    const rawResultRes = await fetch(`/data/results.json?t=${Date.now()}`);
70    const rawResult: RawResult = await rawResultRes.json();
71
72    const coding = [
73      { file: '/data/open_coding/what.json', name: 'What' },
74      { file: '/data/open_coding/why.json', name: 'Why' },
75      { file: '/data/open_coding/fix.json', name: 'Fix' },
76      { file: '/data/open_coding/other.json', name: 'other' },
```

getCompleteResults.ts

No description

```
159    },
160  }));
161
162  const extractCodings = async (responseId: string, codings: Codings): string[]
       => {
       Add a timestamp as query on request to avoid cache issues
163    const codingIdRes = await fetch(`/data/${responseId}/coding.json`);
164    const codingId: string = await codingIdRes.json();
165    const coding = codings.find(({ P }) => P === `P${codingId}`);
166    if (!coding) return [];
167
168    const { P, ...codes } = coding;
169    return Object.entries(codes).reduce<string[]>(
170      (acc, [code, coded]) => (coded.toLowerCase() === 'x') ? [...acc, code] : acc,
```

See correction

Add a comment on the practice

Impacts pour [⋋] packmind

Interface d’affichage d’une pratique

- Mettre en avant la correction

Impacts pour packmind

Interface d’affichage d’une pratique

- Mettre en avant la correction
- Structurer la description

Add a timestamp as query on request to avoid cache issues

Cache

Adding a timestamp as a query parameter in an HTTP request is a common practice utilized to bypass cache issues. This is often done for GET requests to ensure that the server returns a fresh response instead of a cached result. This is achieved by attaching the current timestamp to the URL so that every request URL is unique and isn't matched with a previous cached version.

Pros of this practice :

- It ensures that the response isn't served from the cache but is the latest data from the server.
- The practice can be applied universally in any kind of project or environment.
- It can be very beneficial when real-time data or updates are crucial for the application.

Read more : https://www.section.io/engineering-education/cache-busting/

- Structurer la description

Add a timestamp as query on request to avoid cache issues

Cache

Adding a timestamp as a query parameter in an HTTP request is a common practice utilized to bypass cache issues. This is often done for GET requests to ensure that the server returns a fresh response instead of a cached result. This is achieved by attaching the current timestamp to the URL so that every request URL is unique and isn't matched with a previous cached version.

Pros of this practice :

- It ensures that the response isn't served from the cache but is the latest data from the server.
- The practice can be applied universally in any kind of project or environment.
- It can be very beneficial when real-time data or updates are crucial for the application.

Read more : https://www.section.io/engineering-education/cache-busting/

- Structurer la description

Description

Edit

> Add a timestamp as a query parameter in requests to prevent cache issues.

What is it?

The practice involves appending a unique timestamp to HTTP requests to ensure that responses are not cached by intermediary proxies or browsers. Violating code typically includes API calls or HTTP requests without a timestamp parameter, which might lead to outdated or incorrect data being used.

Why apply it?

Caching can lead to outdated data being used for critical operations. Adding a timestamp ensures that each request is treated as unique, bypassing cache mechanisms and guaranteeing the retrieval of fresh data.

How to Fix it?

To comply, modify the code making HTTP requests to include a timestamp query parameter, such as adding `?timestamp=${Date.now()}` in JavaScript or an equivalent in other languages.

Read more:

https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch#cache_invalidations

Impacts pour packmind

Interface d’affichage d’une pratique

- Mettre en avant la correction
- Structurer la description

Impacts pour packmind

Interface d’affichage d’une pratique

- Mettre en avant la correction
- Structurer la description

Accompagnement des utilisateurs

Les Pratiques de Code : De la Documentation à la Détection

IMT Mines Alès
École Mines-Télécom

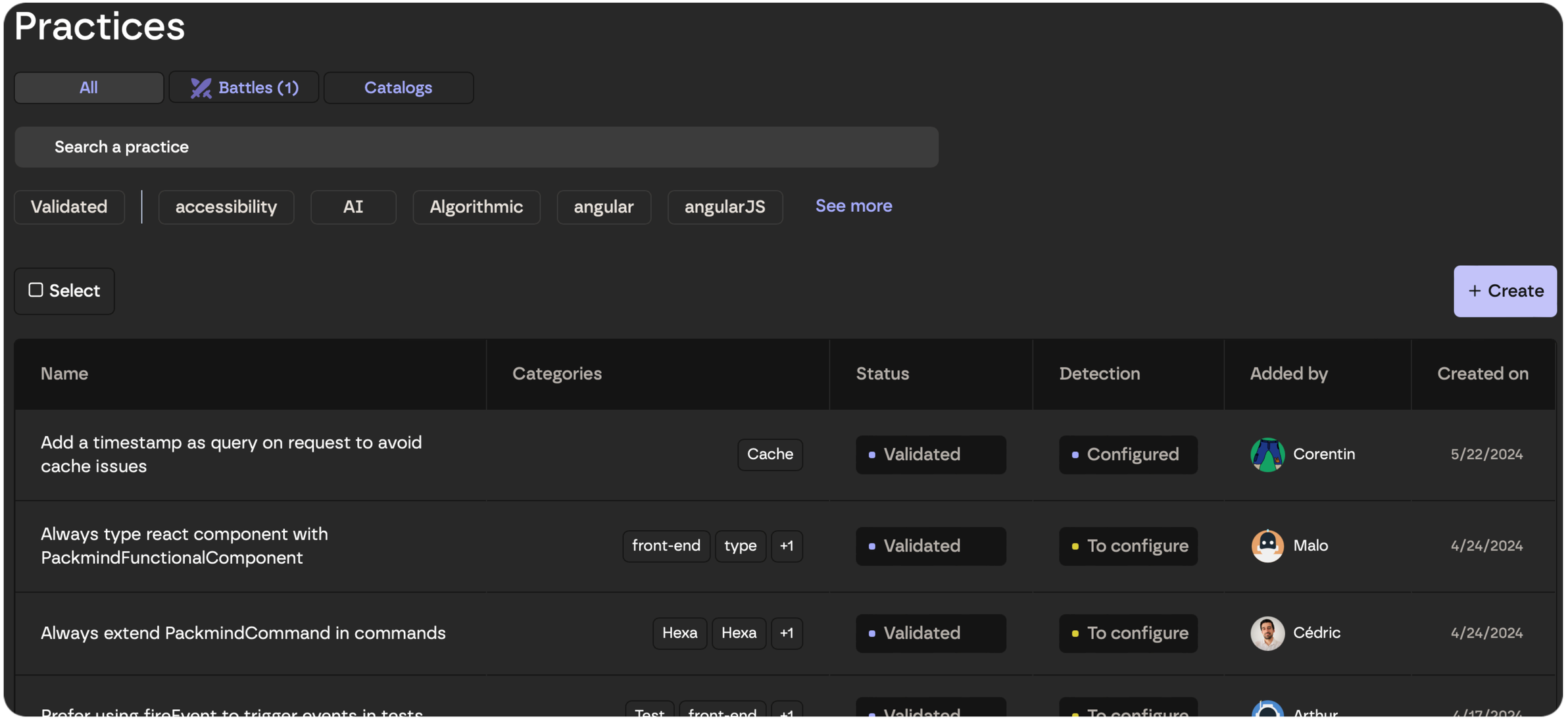
packmind

LaBRI

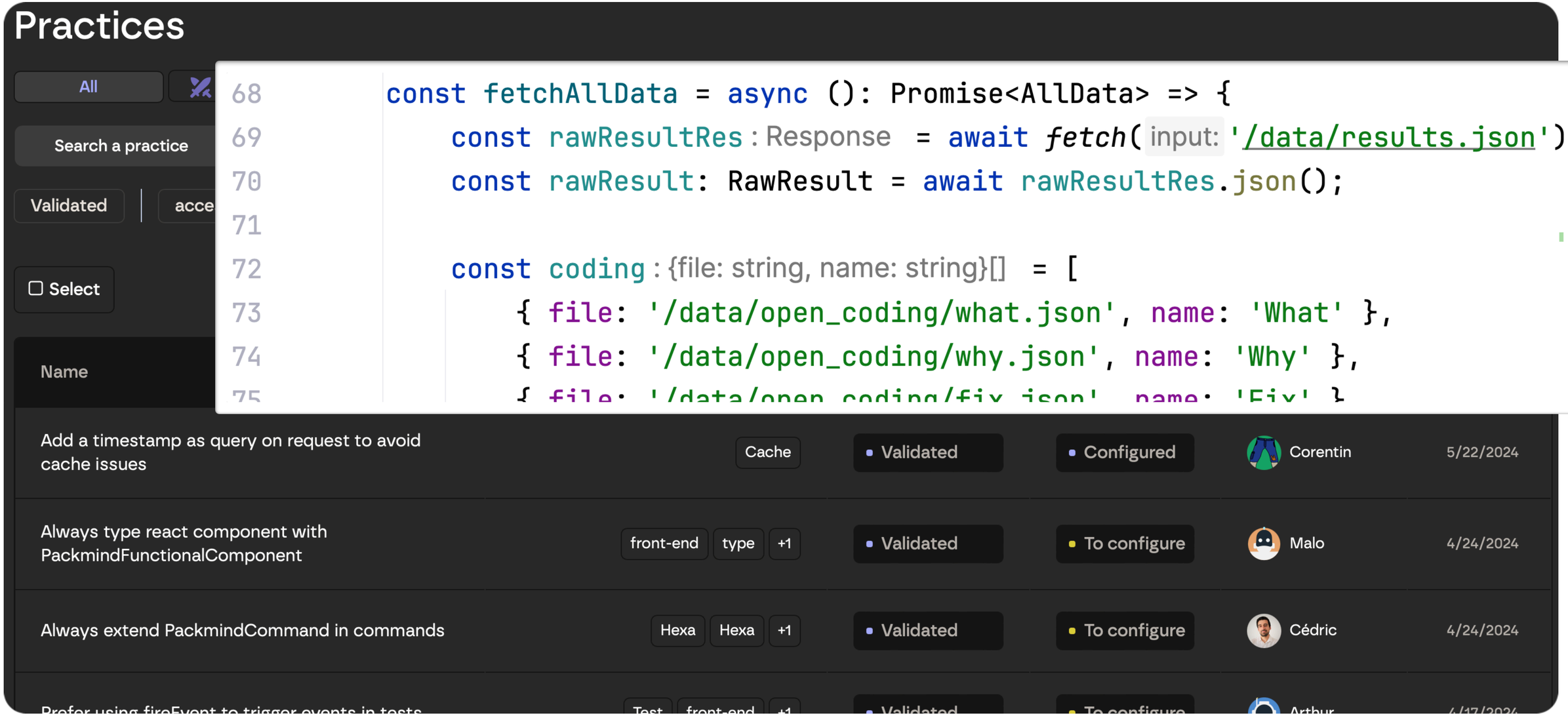
C. Latappy, Q. Perez, T. Degueule, J.-R. Falleri, C. Urtado, S. Vauttier, X. Blanc, et C. Teyton,
« MLinter: Learning Coding Practices from Examples—Dream or Reality? »
in *2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), mars 2023*
doi: 10.1109/SANER56733.2023.00092.

Chez [⋋] packmind

Chez packmind



Chez packmind



Chez [~] packmind

getCompleteResults.ts

No description

```
65    codings: { [key: string]: Codings };
66  };
67
68  const fetchAllData = async (): Promise<AllData> => {
      Add a timestamp as query on request to avoid cache issues
69    const rawResultRes = await fetch(`/data/results.json?t=${Date.now()}`);
70    const rawResult: RawResult = await rawResultRes.json();
71
72    const coding = [
73      { file: '/data/open_coding/what.json', name: 'What' },
74      { file: '/data/open_coding/why.json', name: 'Why' },
75      { file: '/data/open_coding/fix.json', name: 'Fix' },
76      { file: '/data/open_coding/other.json', name: 'other' },
```

getCompleteResults.ts

No description

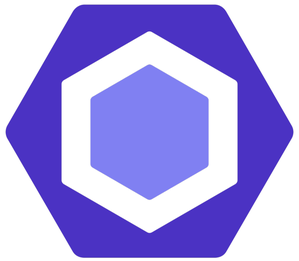
```
159     },
160   }));
161
162   const extractCodings = async (responseId: string, codings: Codings): string[]
      => {
        Add a timestamp as query on request to avoid cache issues
163     const codingIdRes = await fetch(`/data/${responseId}/coding.json`);
164     const codingId: string = await codingIdRes.json();
165     const coding = codings.find(({ P }) => P === `P${codingId}`);
166     if (!coding) return [];
167
168     const { P, ...codes } = coding;
169     return Object.entries(codes).reduce<string[]>(
170       (acc, [code, coded]) => (coded.toLowerCase() === 'x') ? [...acc, code] : acc,
```

See correction

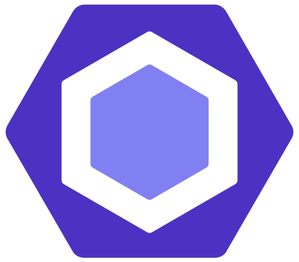
Chez d’autres outils existants

Chez d’autres outils existants

Chez d’autres outils existants



Chez d’autres outils existants



Difficilement extensible

Dans l’État de l’Art

Dans l’État de l’Art

Détection de Code Smells

Dans l’État de l’Art

Détection de Code Smells

Revue systématique sur les différentes approches

Azeem *et al.* 2019

Dans l’État de l’Art

Détection de Code Smells

Revue systématique sur les différentes approches

Azeem *et al.* 2019

20 smells étudiés, avec SVM et DT

Dans l’État de l’Art

Détection de Code Smells

Revue systématique sur les différentes approches

Azeem *et al.* 2019

- 20 smells étudiés, avec SVM et DT
- Entre 50 et 500 instances nécessaires

Dans l’État de l’Art

Détection de Code Smells

Revue systématique sur les différentes approches

Azeem *et al.* 2019

- 20 smells étudiés, avec SVM et DT
- Entre 50 et 500 instances nécessaires

Détection de Pratiques

Dans l’État de l’Art

Détection de Code Smells

Revue systématique sur les différentes approches — Azeem *et al.* 2019

- 20 smells étudiés, avec SVM et DT
- Entre 50 et 500 instances nécessaires

Détection de Pratiques

Détection de bugs en C — Chappelly *et al.* 2017

Dans l'État de l'Art

Détection de Code Smells

Revue systématique sur les différentes approches — Azeem *et al.* 2019

- 20 smells étudiés, avec SVM et DT
- Entre 50 et 500 instances nécessaires

Détection de Pratiques

Détection de bugs en C — Chappelly *et al.* 2017

- 9 bugs étudiés, avec RF et CNN

Dans l’État de l’Art

Détection de Code Smells

Revue systématique sur les différentes approches — Azeem *et al.* 2019

- 20 smells étudiés, avec SVM et DT
- Entre 50 et 500 instances nécessaires

Détection de Pratiques

Détection de bugs en C — Chappelly *et al.* 2017

- 9 bugs étudiés, avec RF et CNN
- 1 000 instances nécessaires

Dans l'État de l'Art

Détection de Code Smells

Revue systématique sur les différentes approches — Azeem *et al.* 2019

- 20 smells étudiés, avec SVM et DT
- Entre 50 et 500 instances nécessaires

Détection de Pratiques

Détection de bugs en C — Chappelly *et al.* 2017

- 9 bugs étudiés, avec RF et CNN
- 1 000 instances nécessaires

Détection de pratiques internes — Ochodek *et al.* 2020

Dans l'État de l'Art

Détection de Code Smells

Revue systématique sur les différentes approches — Azeem *et al.* 2019

- 20 smells étudiés, avec SVM et DT
- Entre 50 et 500 instances nécessaires

Détection de Pratiques

Détection de bugs en C — Chappelly *et al.* 2017

- 9 bugs étudiés, avec RF et CNN
- 1 000 instances nécessaires

Détection de pratiques internes — Ochodek *et al.* 2020

- Utilisation de DT avec tokenisation

Dans l’État de l’Art

Détection de Code Smells

Revue systématique sur les différentes approches — Azeem *et al.* 2019

- 20 smells étudiés, avec SVM et DT
- Entre 50 et 500 instances nécessaires

Détection de Pratiques

Détection de bugs en C — Chappelly *et al.* 2017

- 9 bugs étudiés, avec RF et CNN
- 1 000 instances nécessaires

Détection de pratiques internes — Ochodek *et al.* 2020

- Utilisation de DT avec tokenisation
- 400 instances nécessaires

Questions de Recherche

Apprendre des Pratiques de Code à partir d’Exemples

Questions de Recherche

Apprendre des Pratiques de Code à partir d'Exemples

RQ1: Combien d'exemples sont requis pour apprendre une pratique ?

Questions de Recherche

Apprendre des Pratiques de Code à partir d'Exemples

RQ1: Combien d'exemples sont requis pour apprendre une pratique ?

RQ2: Quelle est la meilleure configuration pour apprendre une pratique ?

MLinter

Design

MLinter

Design

Créer un classifieur binaire pour chaque pratique

MLinter

Design

Créer un classifieur binaire pour chaque pratique

Deux types d’entrée pour apprendre :

MLinter

Design

Créer un classifieur binaire pour chaque pratique

Deux types d’entrée pour apprendre :

- Code non-conforme `let dataLength = data.length;`

MLinter

Design

Créer un classifieur binaire pour chaque pratique

Deux types d’entrée pour apprendre :

- Code non-conforme
- Code conforme

```
let dataLength = data.length;
```

MLinter

Design

Créer un classifieur binaire pour chaque pratique

Deux types d’entrée pour apprendre :

- Code non-conforme `let dataLength = data.length;`
- Code conforme
  - Code fixé `const dataLength = data.length;`

MLinter

Design

Créer un classifieur binaire pour chaque pratique

Deux types d’entrée pour apprendre :

- Code non-conforme
- Code conforme
  - Code fixé
  - Code tiers

```
let dataLength = data.length;
```

```
const dataLength = data.length;
```

```
return data.reduce((acc, value) =>
```

MLinter

CodeBERT

MLinter

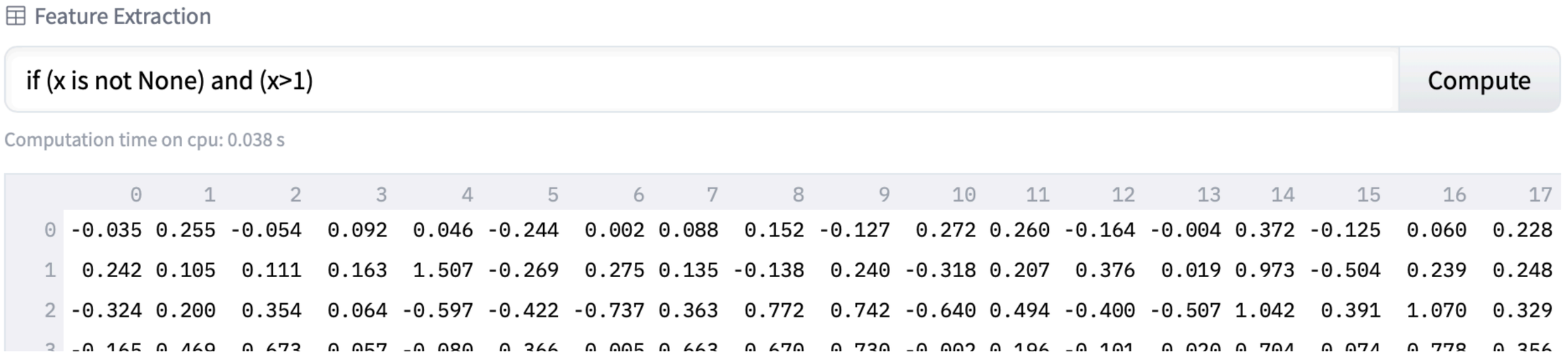
CodeBERT

Modèle pré-entrainé sur 6 langages de programmation

MLinter

CodeBERT

Modèle pré-entrainé sur 6 langages de programmation

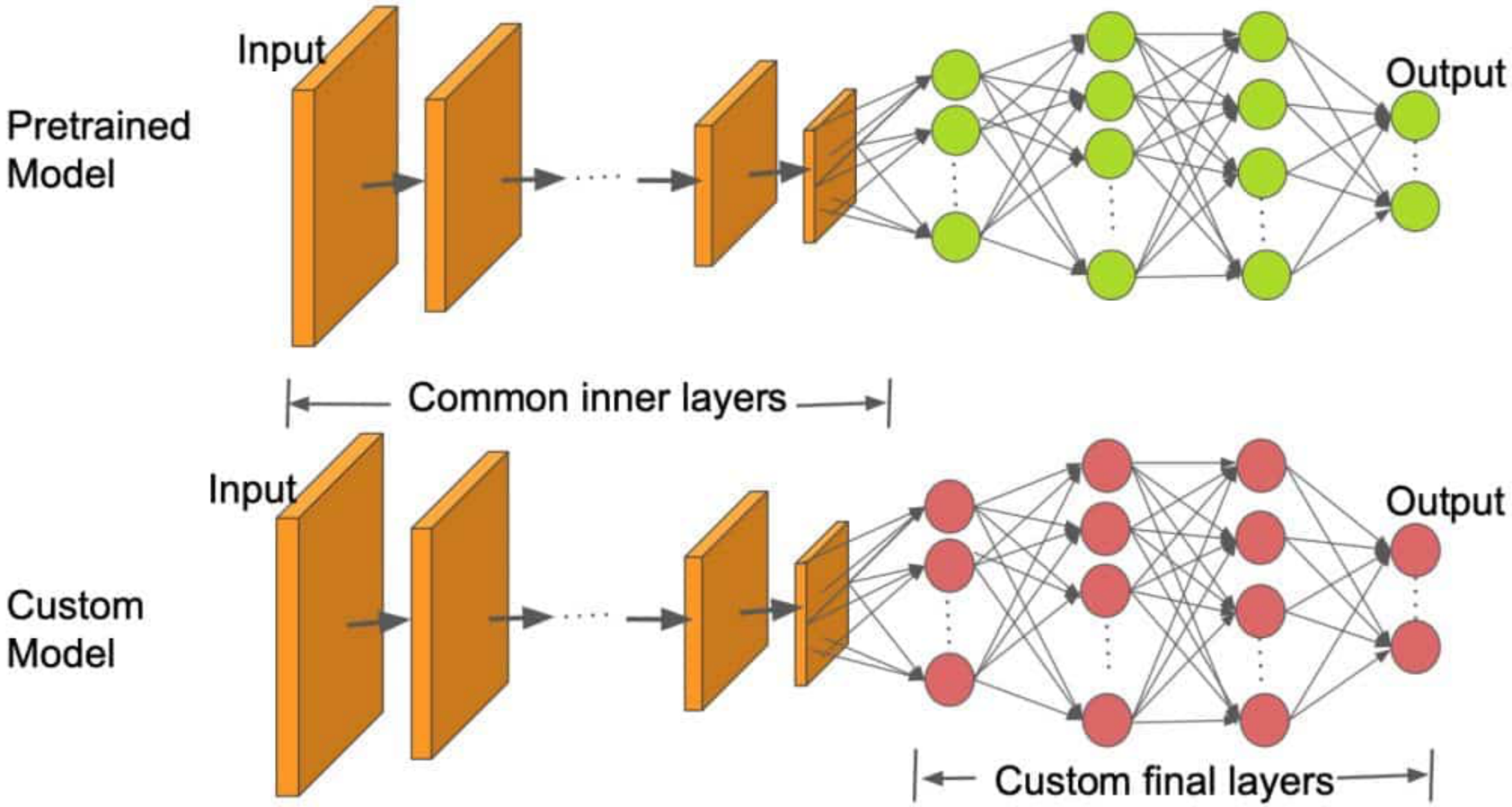


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	-0.035	0.255	-0.054	0.092	0.046	-0.244	0.002	0.088	0.152	-0.127	0.272	0.260	-0.164	-0.004	0.372	-0.125	0.060	0.228
1	0.242	0.105	0.111	0.163	1.507	-0.269	0.275	0.135	-0.138	0.240	-0.318	0.207	0.376	0.019	0.973	-0.504	0.239	0.248
2	-0.324	0.200	0.354	0.064	-0.597	-0.422	-0.737	0.363	0.772	0.742	-0.640	0.494	-0.400	-0.507	1.042	0.391	1.070	0.329
3	-0.165	0.469	0.673	0.057	-0.080	0.366	0.005	0.663	0.670	0.730	-0.002	0.196	-0.101	0.020	0.704	0.074	0.778	0.356

Base CodeBERT output
(https://huggingface.co/microsoft/codebert-base)

MLinter

Fine-tuner CodeBERT pour une nouvelle tâche



Modèle par transfert d'apprentissage
(https://learnopencv.com/image-classification-using-transfer-learning-in-pytorch/)

MLinter

Fine-tuner CodeBERT pour une nouvelle tâche

Code de la pratique ***P***

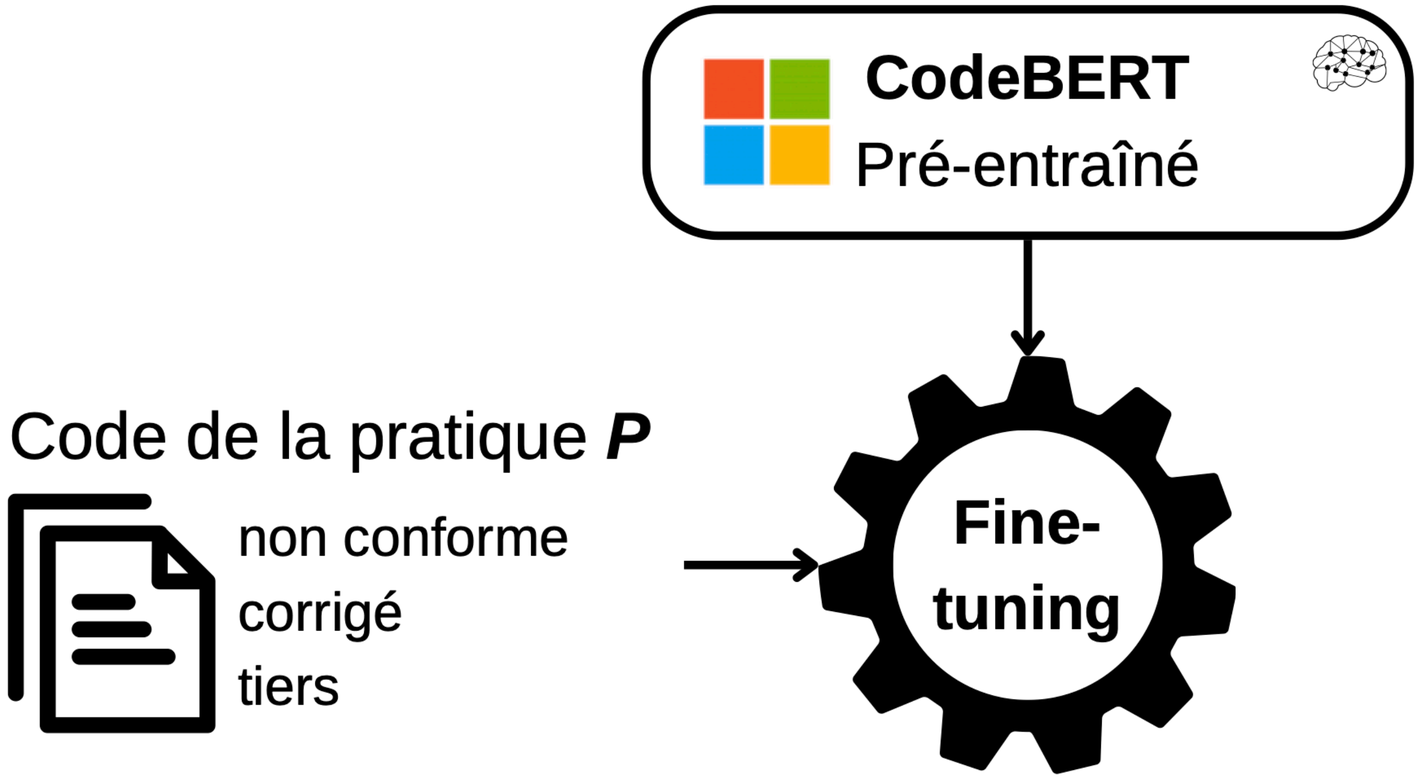
non conforme

corrigé

tiers

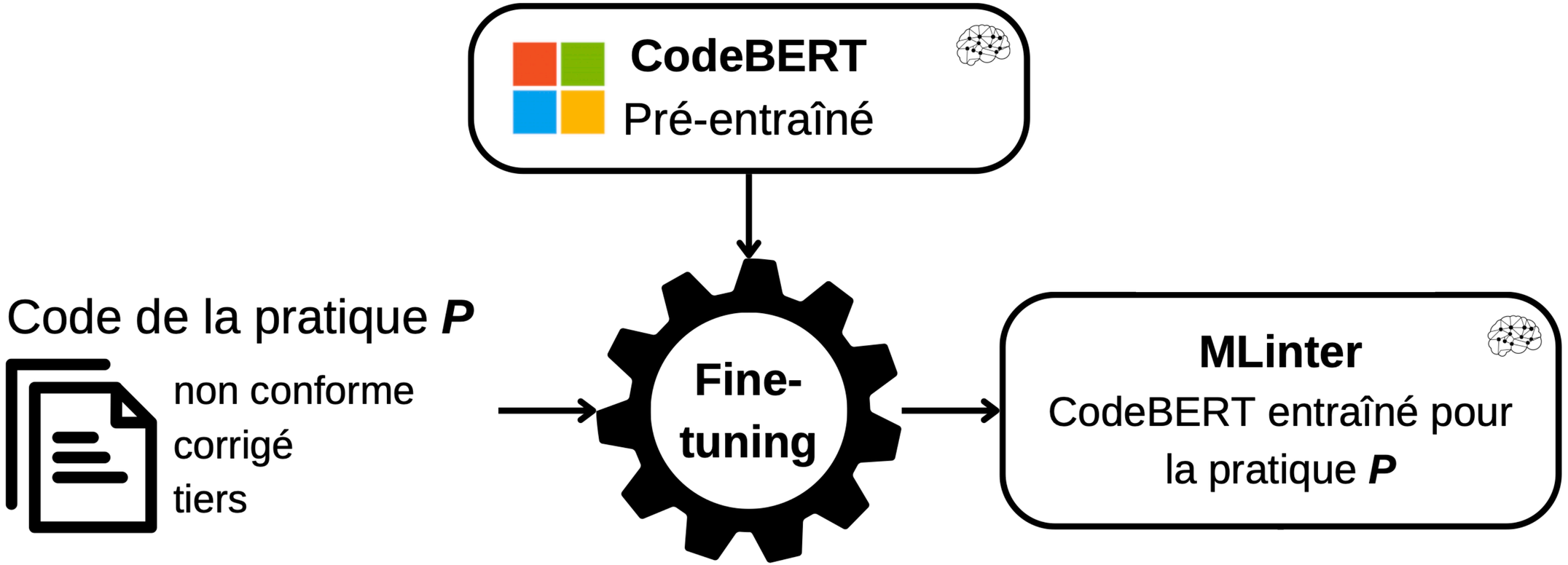
MLinter

Fine-tuner CodeBERT pour une nouvelle tâche



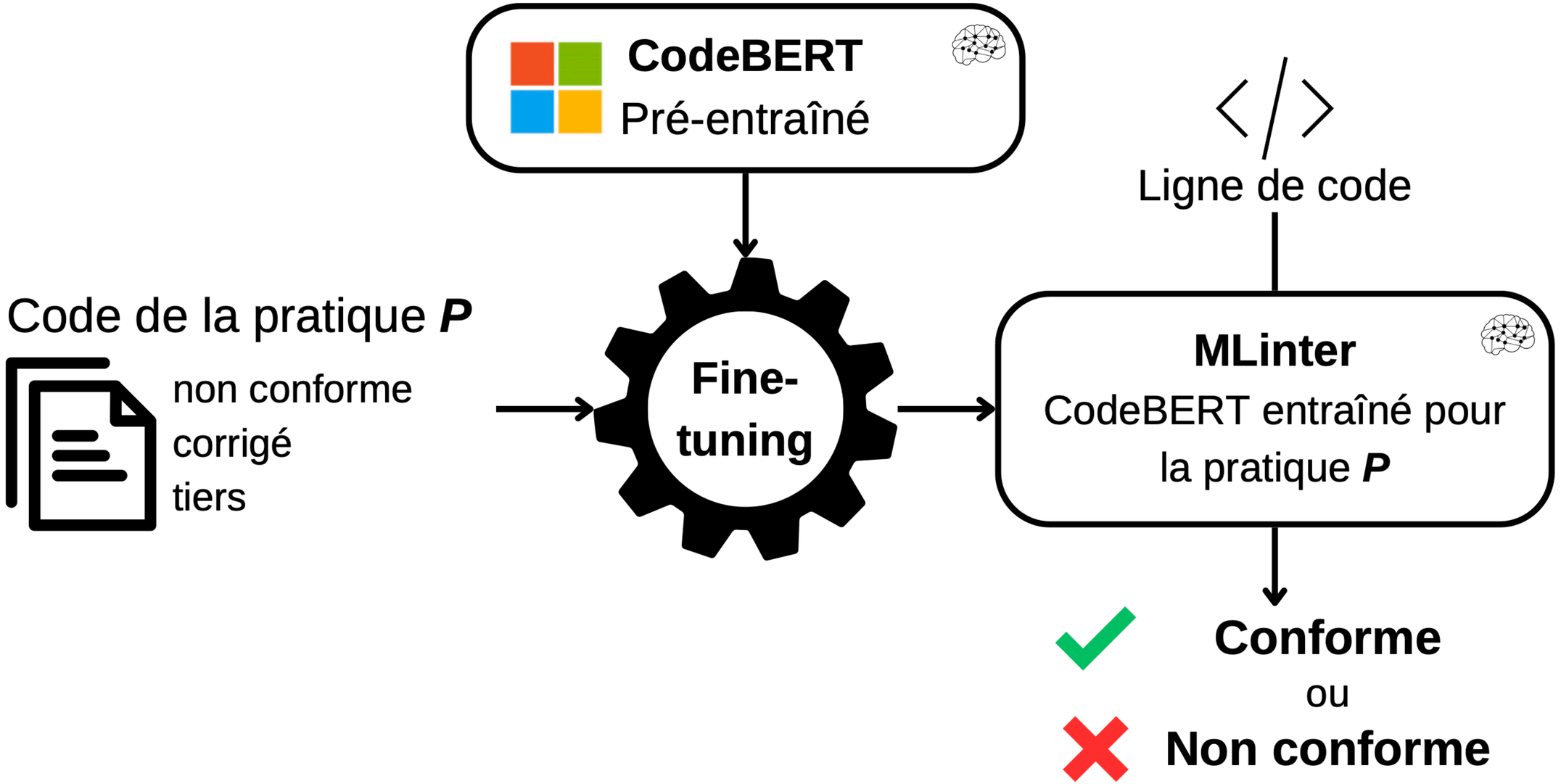
MLinter

Fine-tuner CodeBERT pour une nouvelle tâche

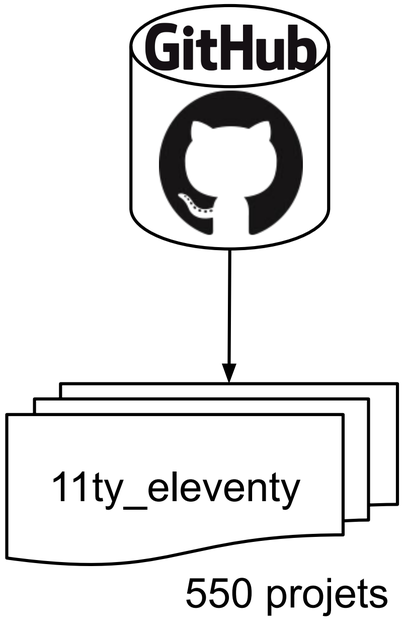


MLinter

Fine-tuner CodeBERT pour une nouvelle tâche



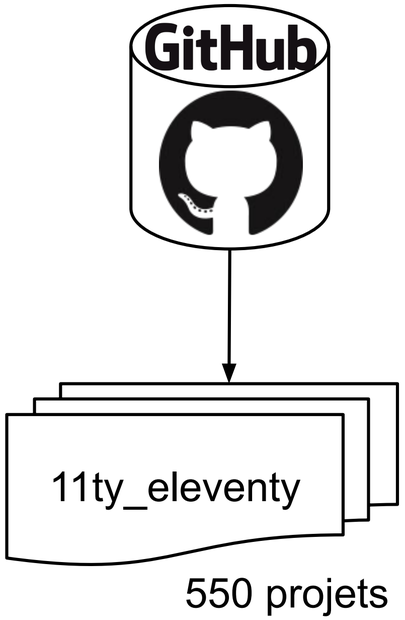
Création de l’ensemble de données



GitHub API pour récupérer tous les projets publics en JavaScript avec plus de 10K ⭐

550 projets clonés

Création de l’ensemble de données



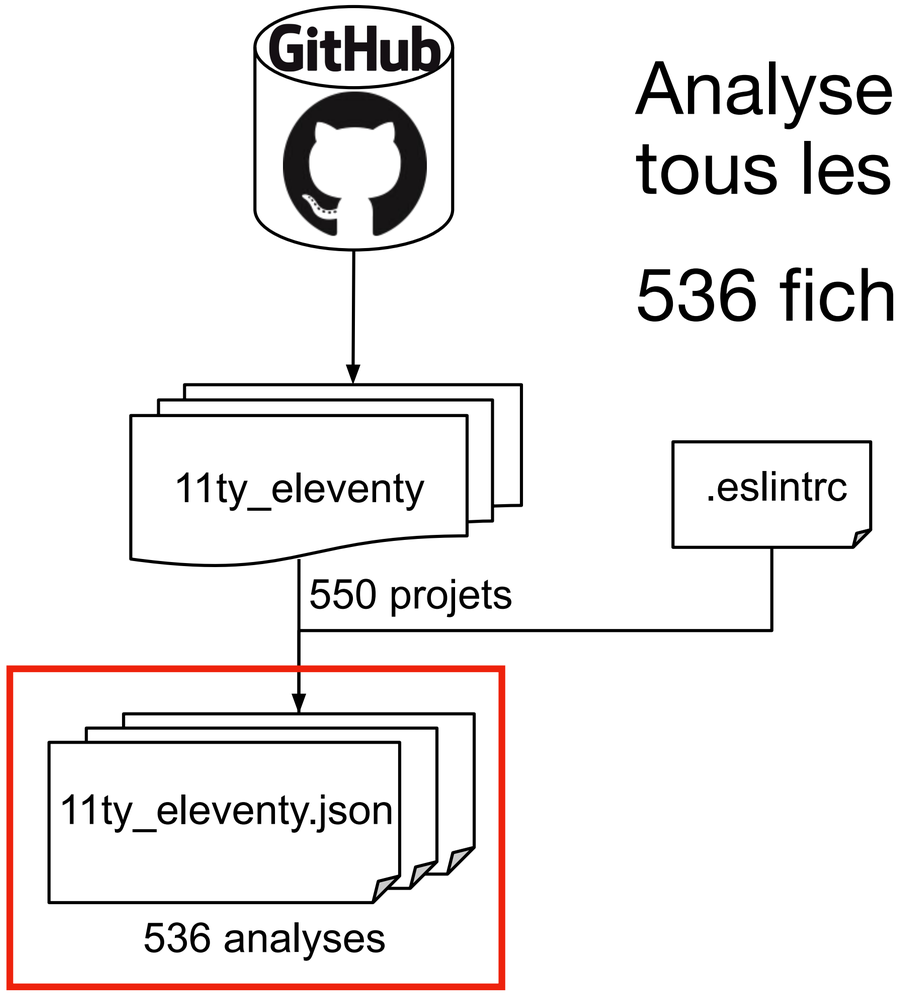
Critères de sélection pour les règles ESLint :

Auto-correction & **Identifiable sur une ligne**

54 règles activées

.eslintrc

Création de l’ensemble de données



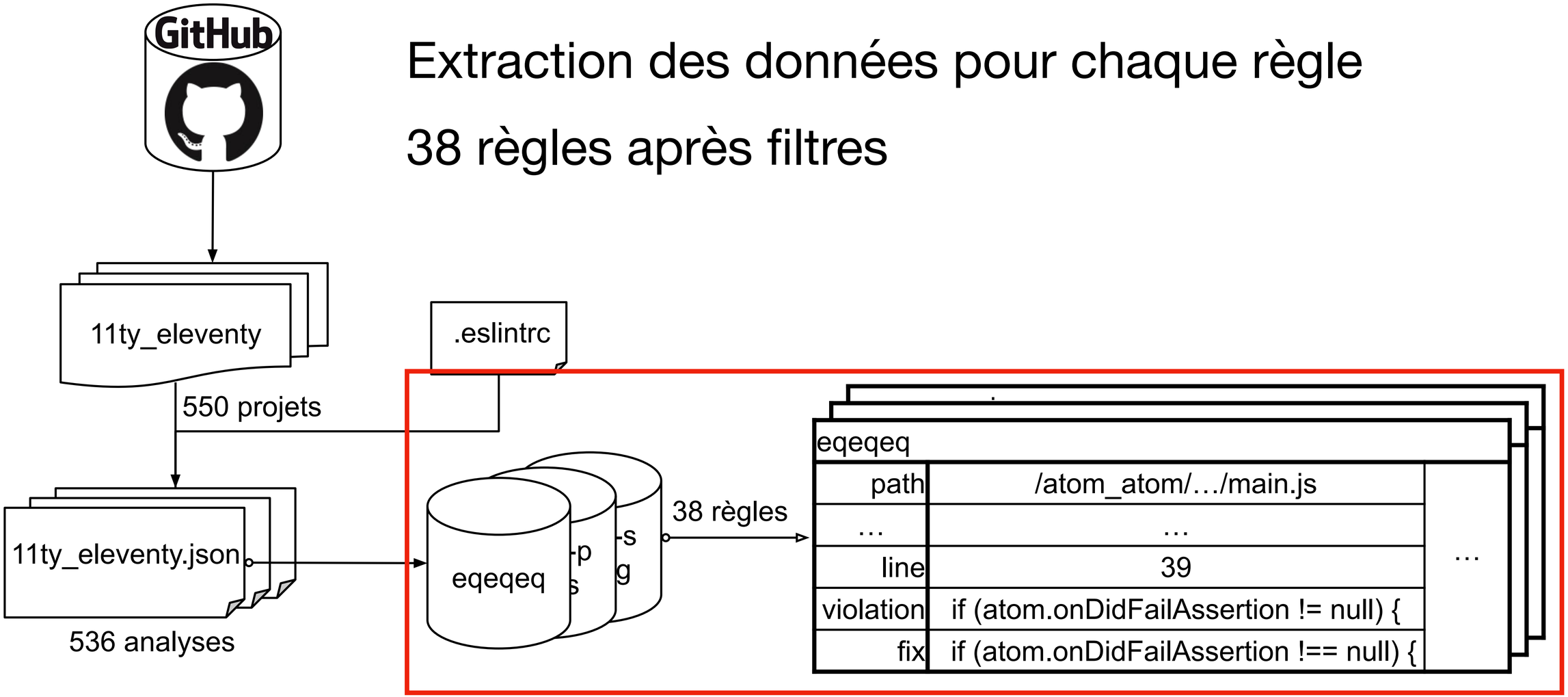
Analyse ESLint avec la configuration réalisée sur tous les projets

536 fichiers d’analyse

Création de l’ensemble de données

Extraction des données pour chaque règle

38 règles après filtres



Création de l’ensemble de données

Création de l’ensemble de données

550 projets

Création de l’ensemble de données

550 projets
218.530 fichiers

Création de l’ensemble de données

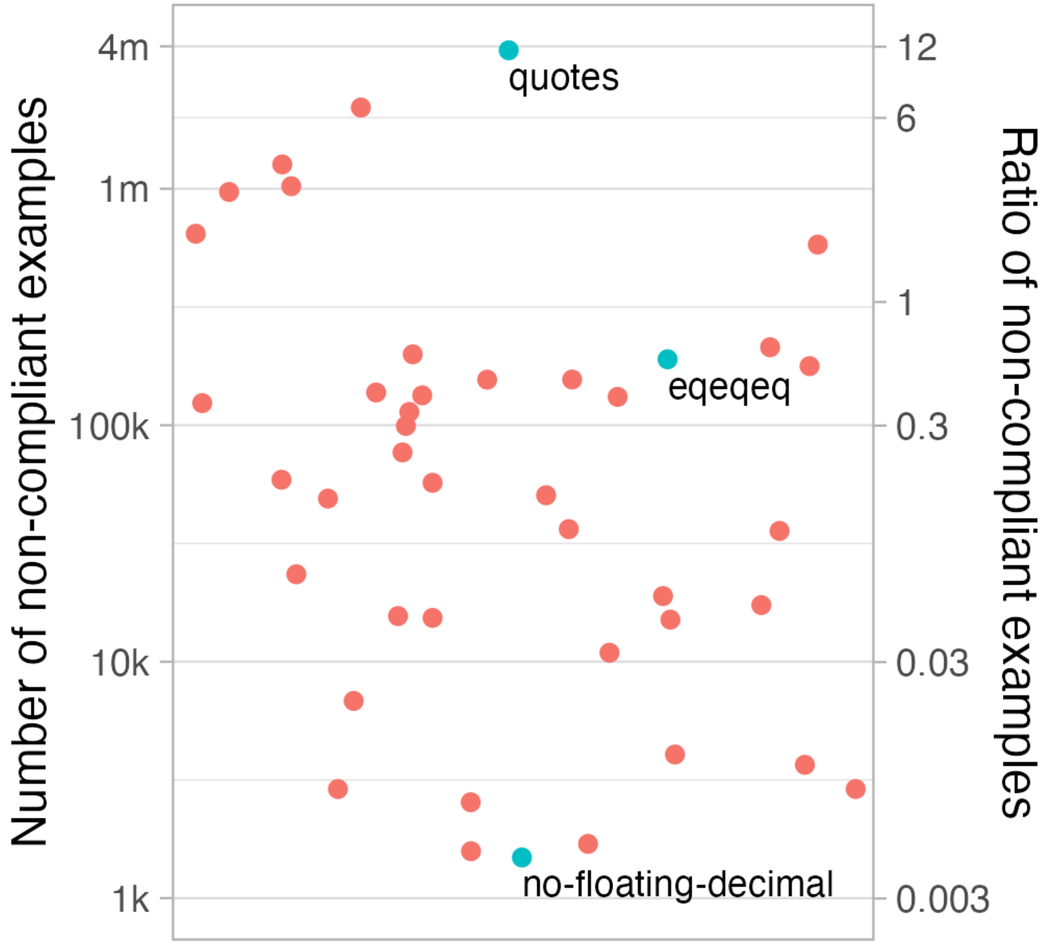
550 projets
218.530 fichiers
+33M lignes de code

Création de l’ensemble de données

550 projets
218.530 fichiers
+33M lignes de code
~13M violations pour 38 règles

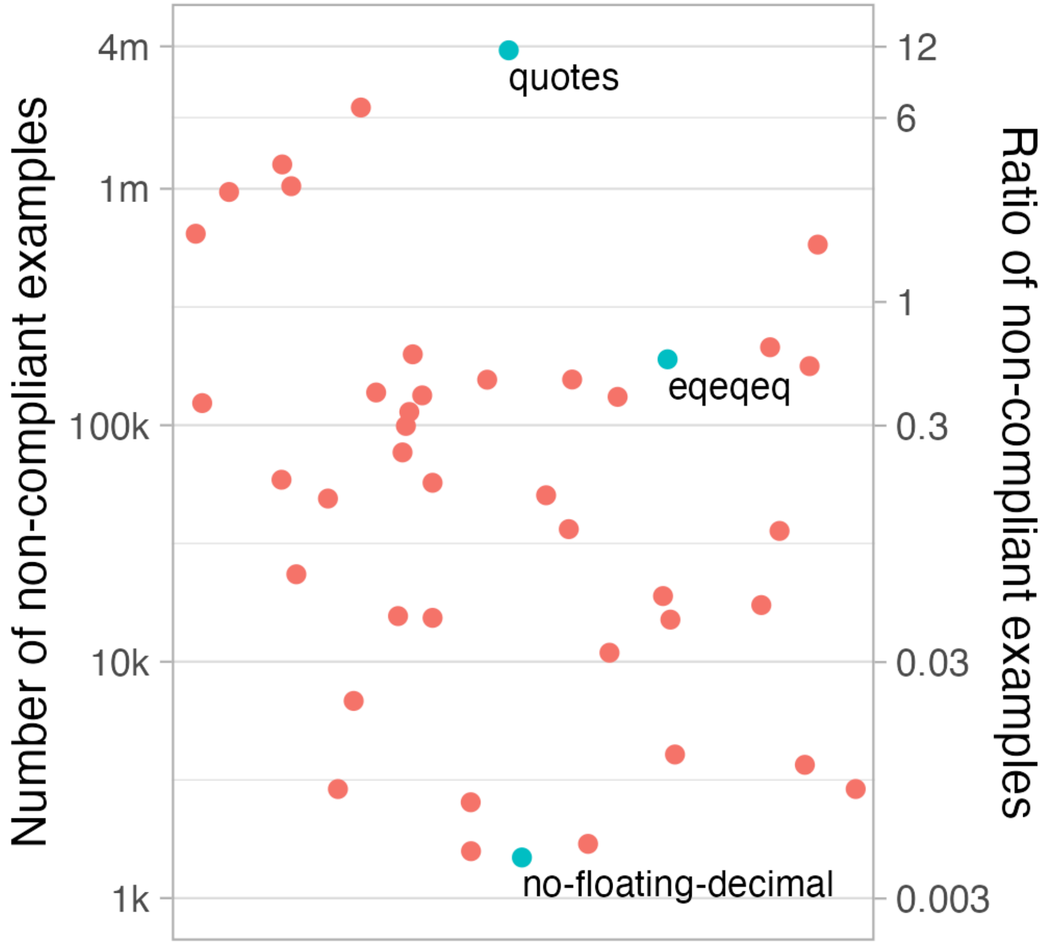
Création de l’ensemble de données

550 projets
218.530 fichiers
+33M lignes de code
~13M violations pour 38 règles



Création de l’ensemble de données

550 projets
218.530 fichiers
+33M lignes de code
~13M violations pour 38 règles



⚠️ Fort déséquilibre entre les règles

Protocole Expérimental

Protocole Expérimental

RQ1 Combien d’exemples sont requis pour apprendre une pratique ?

Protocole Expérimental

RQ1 Combien d'exemples sont requis pour apprendre une pratique ?

3 Tailles d'apprentissage

Protocole Expérimental

RQ1 Combien d'exemples sont requis pour apprendre une pratique ?

3 Tailles d'apprentissage

Petite	10
Moyenne	100
Large	1 000

Protocole Expérimental

RQ2 Quelle est la meilleure configuration pour apprendre une pratique ?

Protocole Expérimental

RQ2 Quelle est la meilleure configuration pour apprendre une pratique ?

Non conforme (50%) **Conforme (50%)**

Protocole Expérimental

RQ2 Quelle est la meilleure configuration pour apprendre une pratique ?

Non conforme (50%)	**Conforme (50%)**
Violation (50%)	

Protocole Expérimental

<u>RQ2</u> Quelle est la meilleure configuration pour apprendre une pratique ?

<table>
<tr><th>Non conforme (50%)</th><th>Conforme (50%)</th></tr>
<tr><td rowspan="3">Violation (50%)</td><td>Tiers (50%)</td></tr>
<tr><td>Fixé (50%)</td></tr>
<tr><td>Fixé (25%)</td><td>Tiers (25%)</td></tr>
</table>

Protocole Expérimental

RQ2 Quelle est la meilleure configuration pour apprendre une pratique ?

<table>
<tr><th></th><th>Non conforme (50%)</th><th colspan="2">Conforme (50%)</th></tr>
<tr><td>VE</td><td rowspan="3">Violation (50%)</td><td colspan="2">Tiers (50%)</td></tr>
<tr><td>VF</td><td colspan="2">Fixé (50%)</td></tr>
<tr><td>VFE</td><td>Fixé (25%)</td><td>Tiers (25%)</td></tr>
</table>

Protocole Expérimental

	10 (S)	**100 (M)**	**1 000 (L)**
VE	S / VE	M / VE	L / VE
VF	S / VF	M / VF	L / VF
VFE	S / VFE	M / VFE	L / VFE

9 configurations pour chaque pratique

Protocole Expérimental

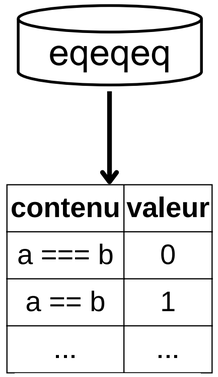
Pour chaque practique ***P*** (eqeqeq) et chaque configuration ***C*** (M/VF)

Protocole Expérimental

Pour chaque practique ***P*** (eqeqeq) et chaque configuration ***C*** (M/VF)

Protocole Expérimental

Pour chaque practique ***P*** (eqeqeq) et chaque configuration ***C*** (M/VF)

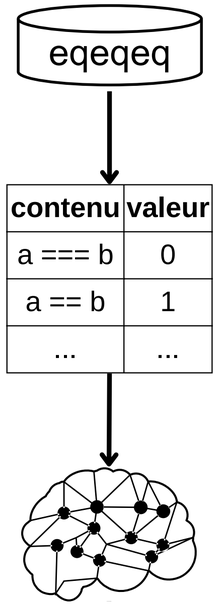


contenu	valeur
a === b	0
a == b	1
...	...

Création d’un ensemble d’apprentissage aléatoire pour ***P*** en respectant ***C***

Protocole Expérimental

Pour chaque practique ***P*** (eqeqeq) et chaque configuration ***C*** (M/VF)

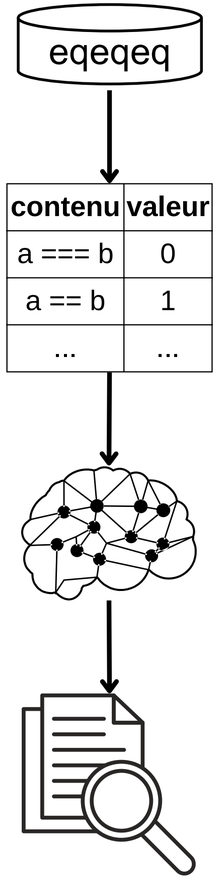


Création d'un ensemble d'apprentissage aléatoire pour ***P*** en respectant ***C***

Fine-tuning de CodeBERT pour obtenir un nouveau classifieur

Protocole Expérimental

Pour chaque practique ***P*** (eqeqeq) et chaque configuration ***C*** (M/VF)



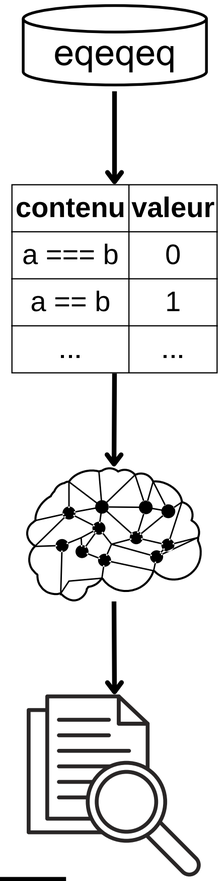
Création d'un ensemble d'apprentissage aléatoire pour ***P*** en respectant ***C***

Fine-tuning de CodeBERT pour obtenir un nouveau classifieur

Évaluation du classifieur en mesurant Précision et Rappel

Protocole Expérimental

Pour chaque practique ***P*** (eqeqeq) et chaque configuration ***C*** (M/VF)



Création d'un ensemble d'apprentissage aléatoire pour ***P*** en respectant ***C***

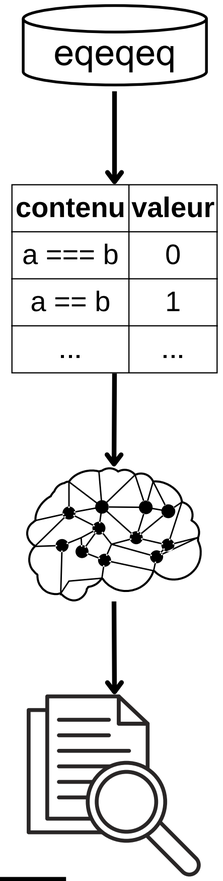
Fine-tuning de CodeBERT pour obtenir un nouveau classifieur

Évaluation du classifieur en mesurant Précision et Rappel

x 100

Protocole Expérimental

Pour chaque practique ***P*** (eqeqeq) et chaque configuration ***C*** (M/VF)



Création d'un ensemble d'apprentissage aléatoire pour ***P*** en respectant ***C***

Fine-tuning de CodeBERT pour obtenir un nouveau classifieur

Évaluation du classifieur en mesurant Précision et Rappel

x 100

34 200 classifieurs entrainés

Protocole Expérimental

Validation équilibrée

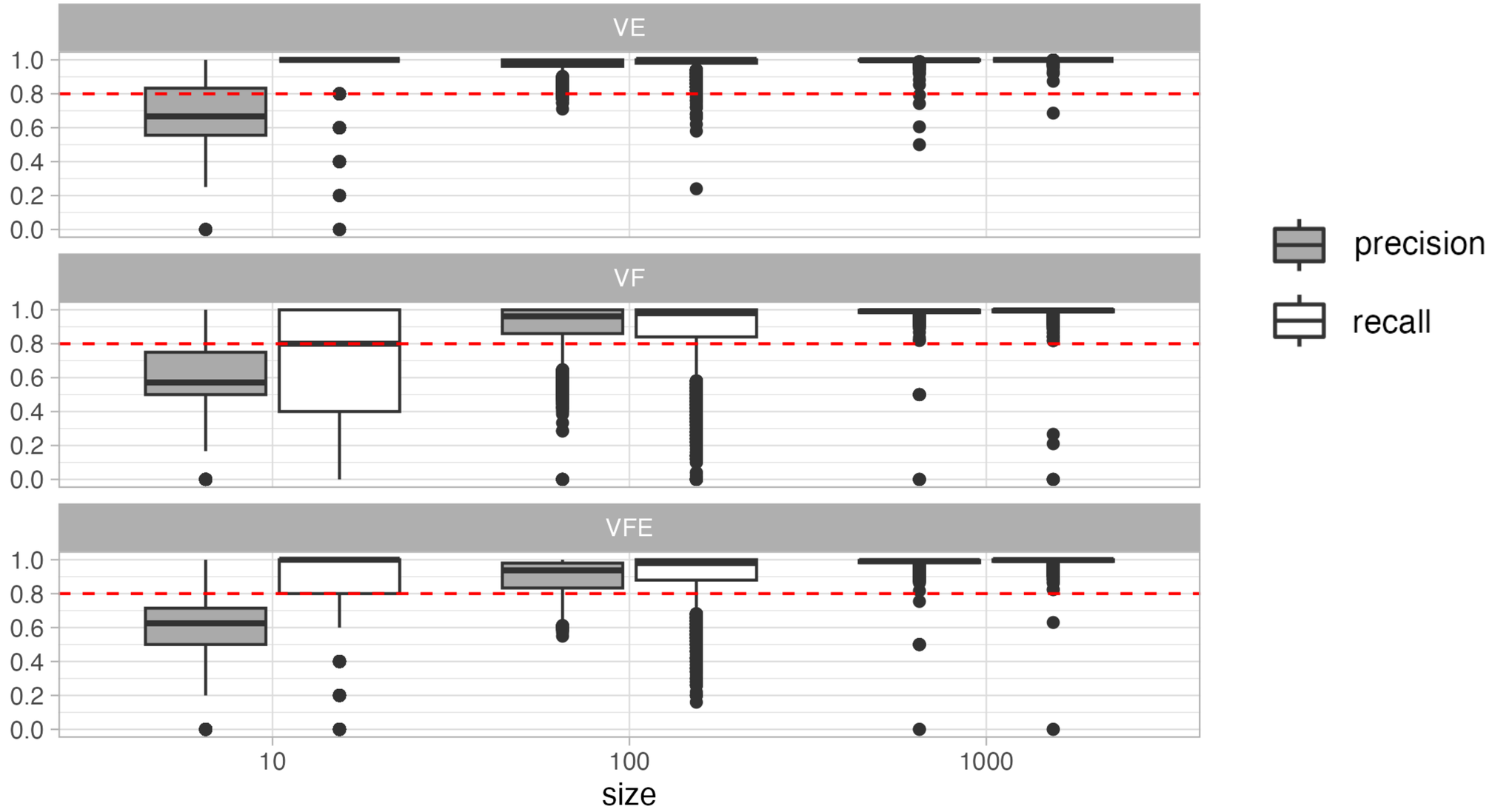
Ensemble de même taille et équilibre que l’entraînement

Validation réaliste

Ensemble avec un équilibre similaire à du vrai code source

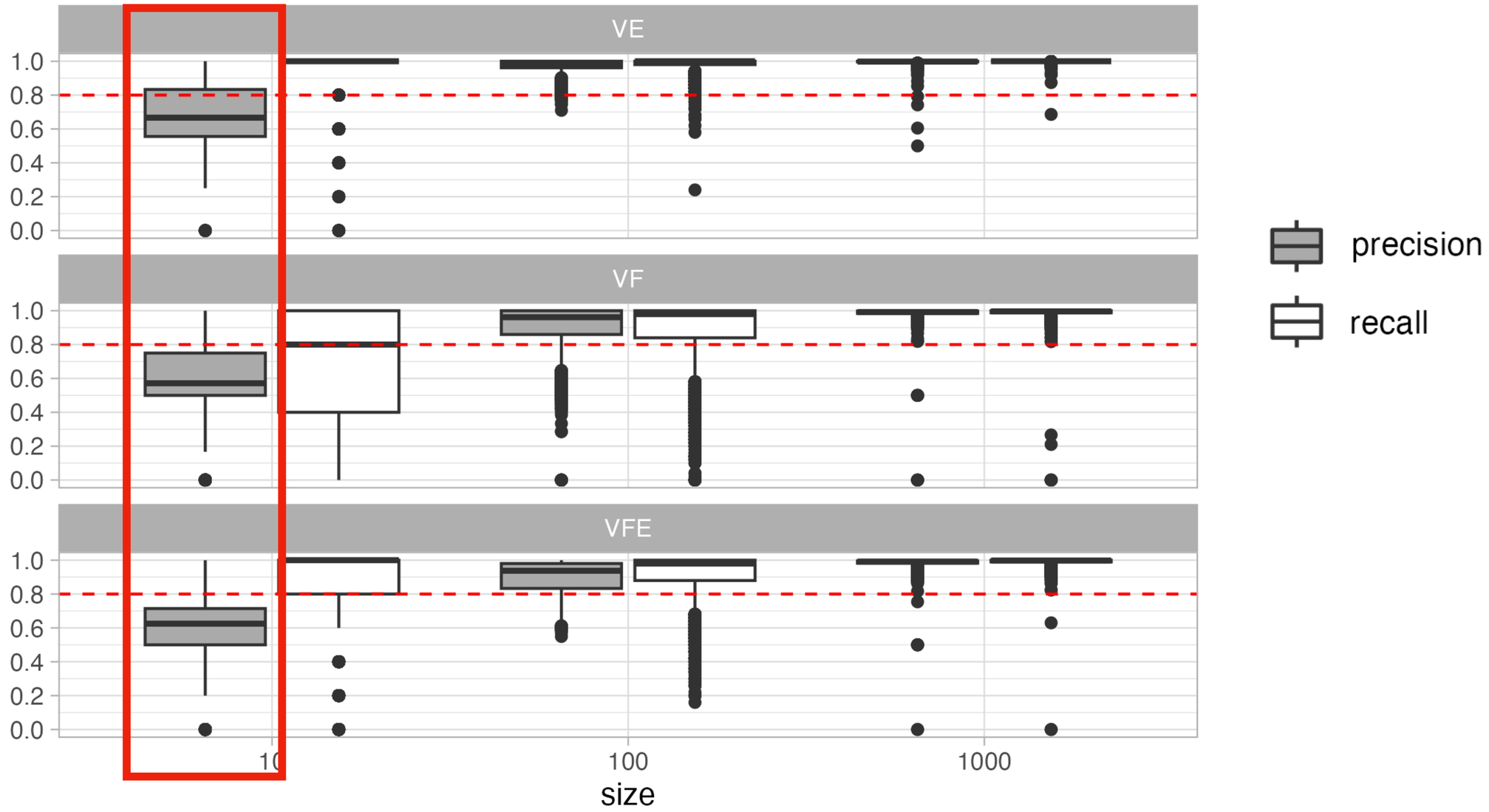
Résultats

Validation équilibrée



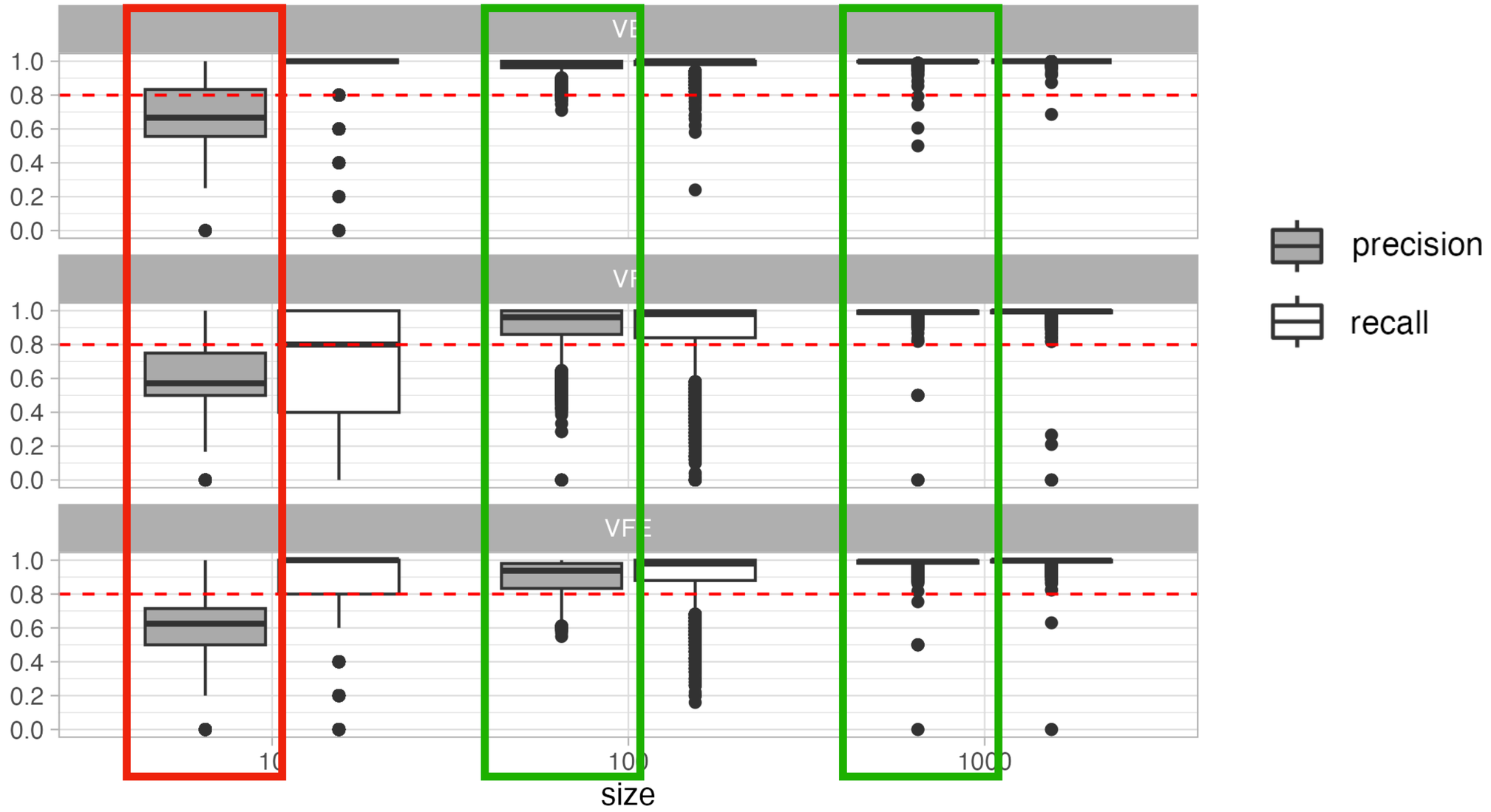
Résultats

Validation équilibrée



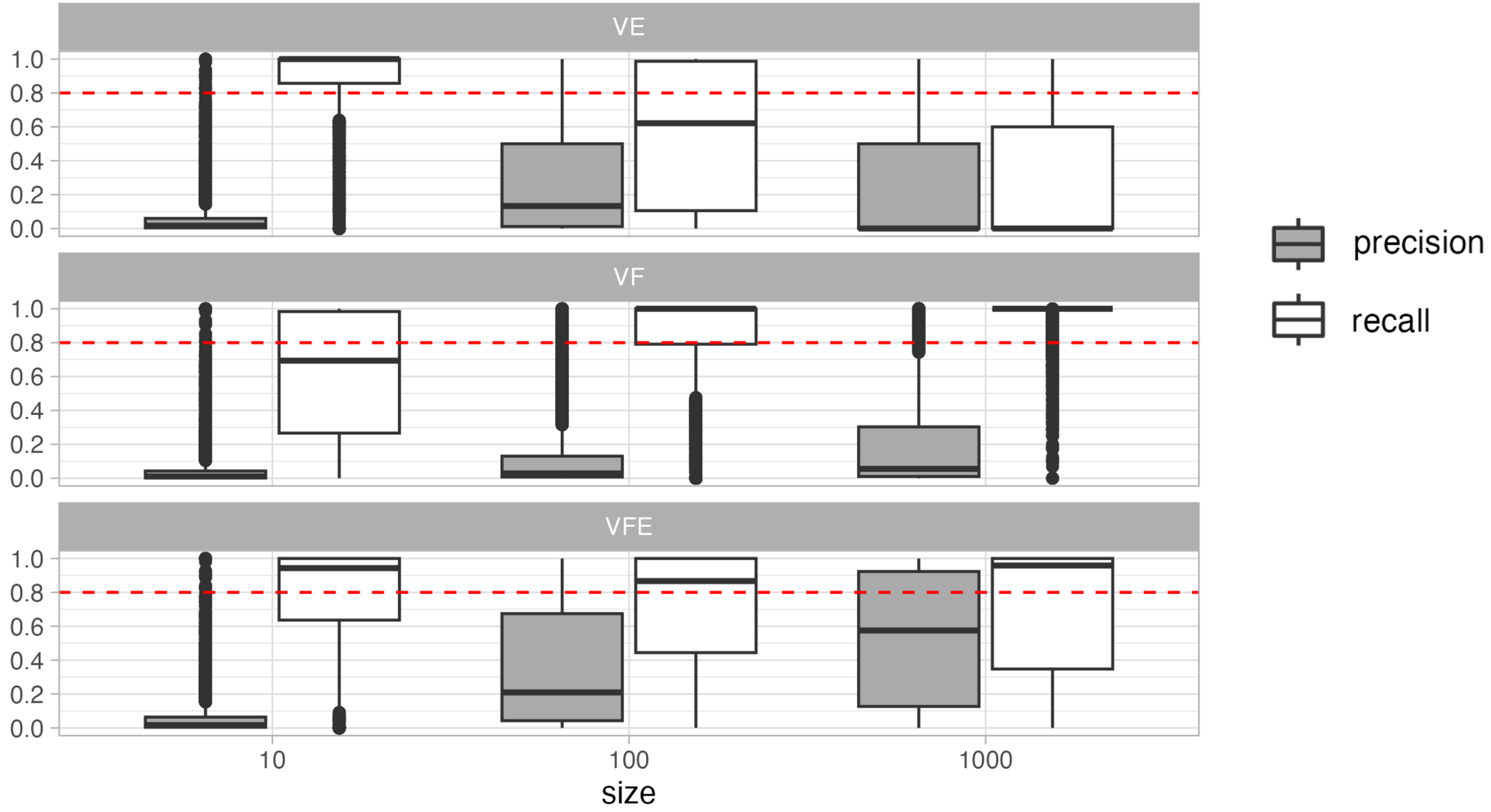
Résultats

Validation équilibrée



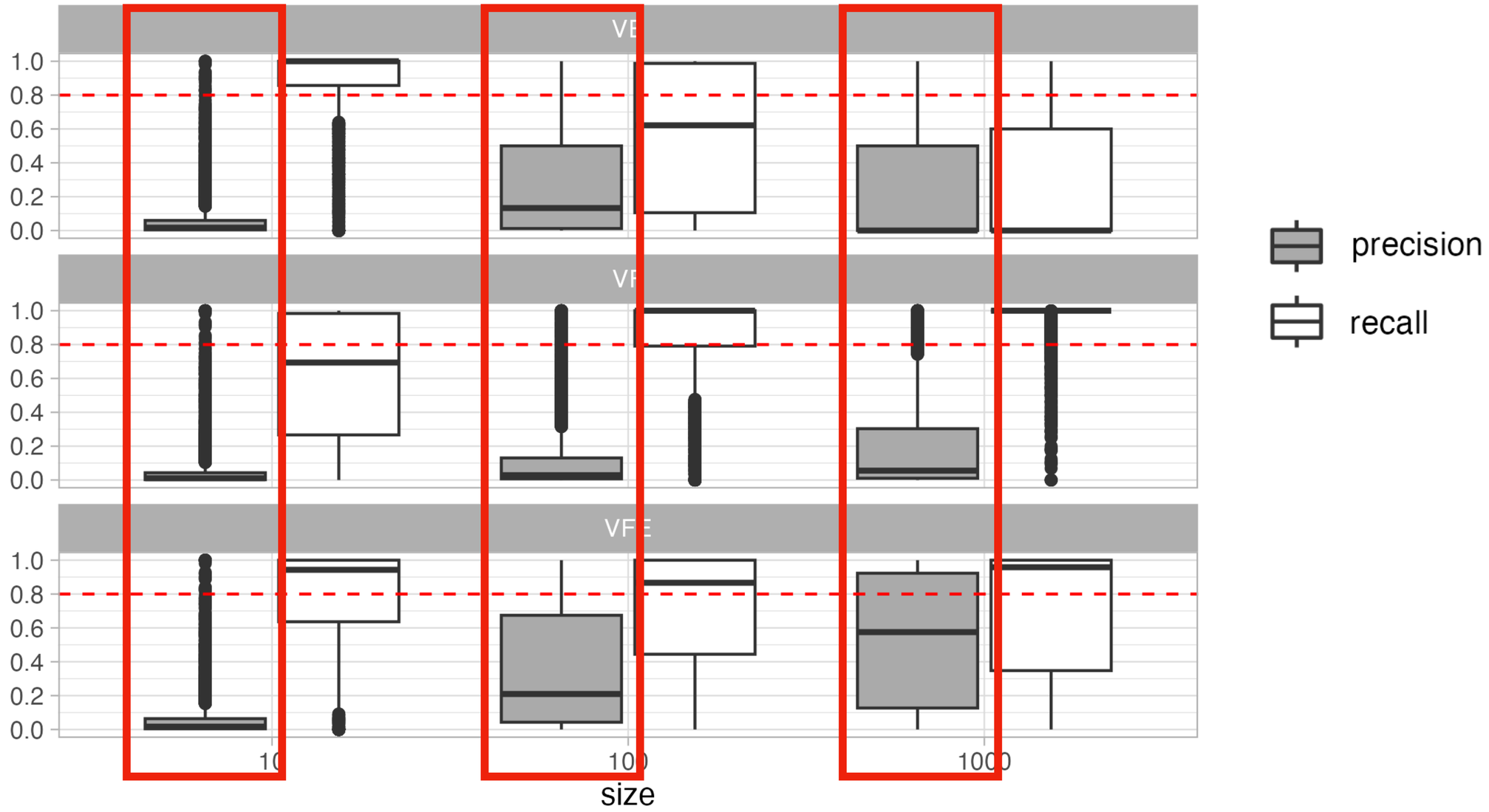
Résultats

Validation réaliste



Résultats

Validation réaliste



Résultats

Conclusion

Résultats

Conclusion

RQ1:

Résultats

Conclusion

RQ1:

- Performances augmentent avec la taille d’apprentissage

Résultats

Conclusion

RQ1:

- Performances augmentent avec la taille d’apprentissage
- Taille 10 clairement insuffisante

Résultats

Conclusion

RQ1:

- Performances augmentent avec la taille d’apprentissage
- Taille 10 clairement insuffisante
- Taille 100 similaire à la taille 1 000

Résultats

Conclusion

RQ1:

- Performances augmentent avec la taille d’apprentissage
- Taille 10 clairement insuffisante
- Taille 100 similaire à la taille 1 000

RQ2:

Résultats

Conclusion

RQ1:

- Performances augmentent avec la taille d’apprentissage
- Taille 10 clairement insuffisante
- Taille 100 similaire à la taille 1 000

RQ2:

- Aucune configuration n’émerge

En résumé

Questions de Recherche

Apprendre des Pratiques de Code à partir d'Exemples

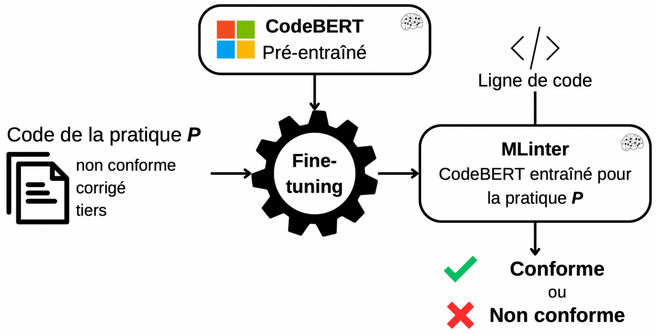
RQ1: Combien d'exemples sont requis pour apprendre une pratique ?

RQ2: Quelle est la meilleure configuration pour apprendre une pratique ?

34

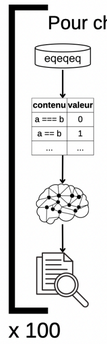
MLinter

Fine-tuner CodeBERT pour une nouvelle tâche



38

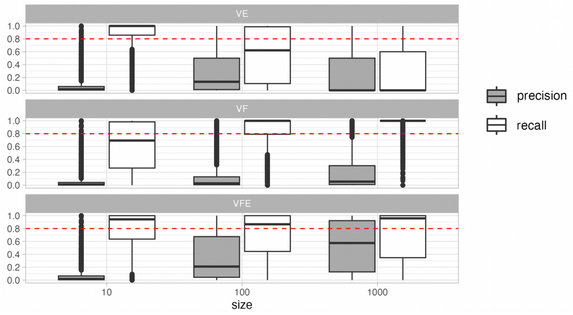
Protocole Expérimental



47

Résultats

Validation réaliste



50

Résultats

Conclusion

RQ1:

- Résultats augmentent avec la taille d'apprentissage
- Taille 10 clairement insuffisante
- Taille 100 similaire à la taille 1 000

RQ2:

- Aucune configuration émerge

51

Impacts pour [⋋] packmind

Impacts pour packmind

MLinter non applicable

Impacts pour [⋋] packmind

MLinter non applicable

Cependant…

Impacts pour packmind

```
68      const fetchAllData = async (): Promise<AllData> => {
69          const rawResultRes : Response  = await fetch( input: '/data/results.json');
70          const rawResult: RawResult = await rawResultRes.json();
71
72          const coding : {file: string, name: string}[]  = [
73              { file: '/data/open_coding/what.json', name: 'What' },
74              { file: '/data/open_coding/why.json', name: 'Why' },
75              { file: '/data/open_coding/fix.json', name: 'Fix' },
```

Impacts pour [⋋] packmind

```
68      const fetchAllData = async (): Promise<AllData> => {
69          Add a timestamp as query on request to avoid cache issues ( input: '/data/results.json');
70          const rawResult: RawResult = await rawResultRes.json();
71
72          const coding : {file: string, name: string}[]  = [
73              { file: '/data/open_coding/what.json', name: 'What' },
74              { file: '/data/open_coding/why.json', name: 'Why' },
75              { file: '/data/open_coding/fix.json', name: 'Fix' },
```

Perspectives

Perspectives

Perspectives

Détecter les violations de pratiques internes

Perspectives

Détecter les violations de pratiques internes

LLMLinter: Learning Coding Practices from Examples—Another Dream?

Perspectives

Détecter les violations de pratiques internes

Améliorer la qualité de la documentation des pratiques

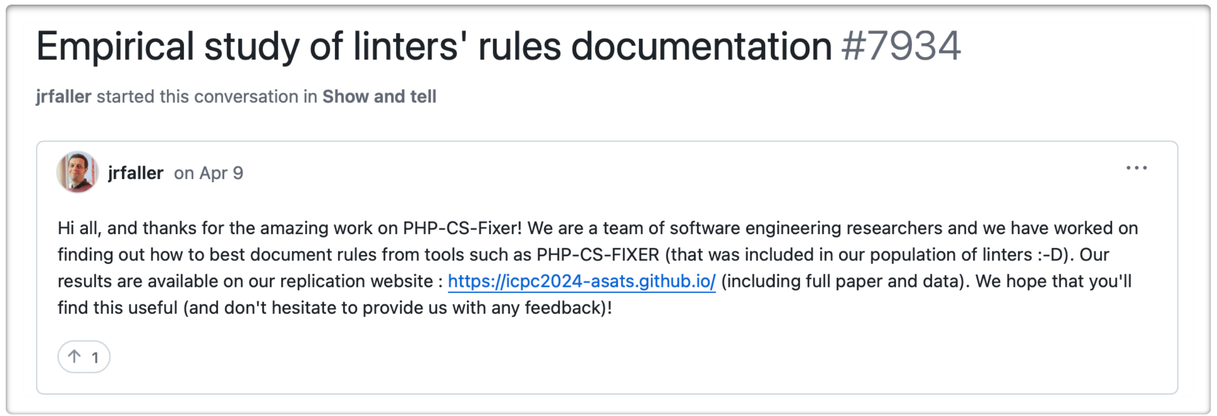
LLMLinter: Learning Coding Practices from Examples—Another Dream?

Perspectives

Détecter les violations de pratiques internes

LLMLinter: Learning Coding Practices from Examples—Another Dream?

Améliorer la qualité de la documentation des pratiques



Empirical study of linters' rules documentation #7934

jrfaller started this conversation in **Show and tell**

jrfaller on Apr 9

Hi all, and thanks for the amazing work on PHP-CS-Fixer! We are a team of software engineering researchers and we have worked on finding out how to best document rules from tools such as PHP-CS-FIXER (that was included in our population of linters :-D). Our results are available on our replication website : https://icpc2024-asats.github.io/ (including full paper and data). We hope that you'll find this useful (and don't hesitate to provide us with any feedback)!

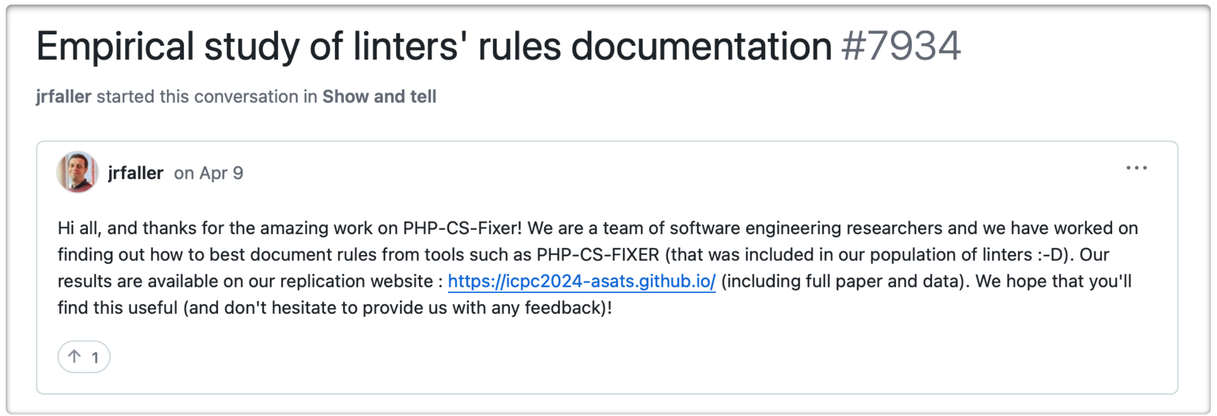
↑ 1

Perspectives

Détecter les violations de pratiques internes

LLMLinter: Learning Coding Practices from Examples—Another Dream?

Améliorer la qualité de la documentation des pratiques



Inférer Automatiquement une Correction

Perspectives

Détecter les violations de pratiques internes

LLMLinter: Learning Coding Practices from Examples—Another Dream?

Améliorer la qualité de la documentation des pratiques

Empirical study of linters' rules documentation #7934

jrfaller started this conversation in **Show and tell**

jrfaller on Apr 9

Hi all, and thanks for the amazing work on PHP-CS-Fixer! We are a team of software engineering researchers and we have worked on finding out how to best document rules from tools such as PHP-CS-FIXER (that was included in our population of linters :-D). Our results are available on our replication website : https://icpc2024-asats.github.io/ (including full paper and data). We hope that you'll find this useful (and don't hesitate to provide us with any feedback)!

↑ 1

Inférer Automatiquement une Correction

```
fetch( input: '/data/results.json');
```

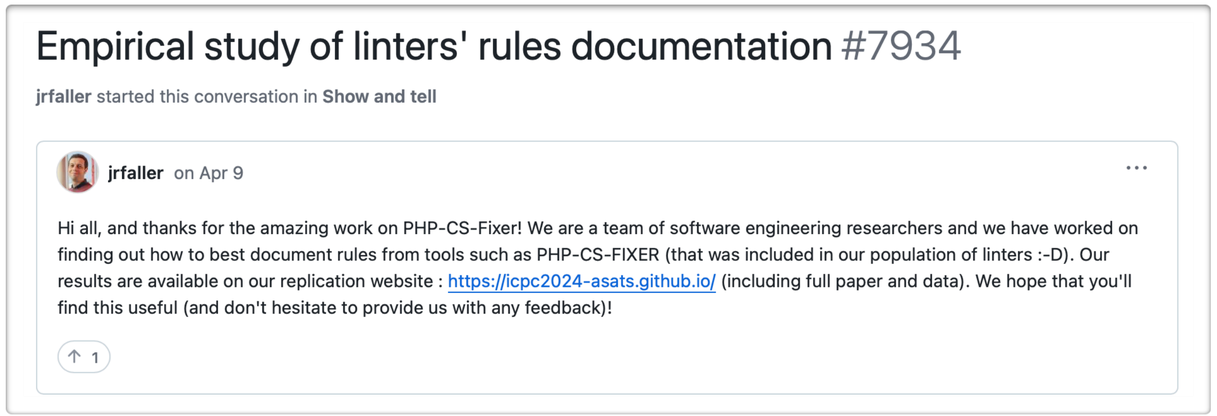
```
fetch( input: `/data/results.json?t=${ Date.now() }`);
```

Perspectives

Détecter les violations de pratiques internes

LLMLinter: Learning Coding Practices from Examples—Another Dream?

Améliorer la qualité de la documentation des pratiques



Inférer Automatiquement une Correction

```
fetch( input: '/data/results.json');
```

```
fetch( input: `/data/results.json?t=${ Date.now() }`);
```

Maintenir à Jour la Base de Connaissances

Perspectives

Détecter les violations de pratiques internes

LLMLinter: Learning Coding Practices from Examples—Another Dream?

Améliorer la qualité de la documentation des pratiques

Empirical study of linters' rules documentation #7934

jrfaller started this conversation in **Show and tell**

jrfaller on Apr 9

Hi all, and thanks for the amazing work on PHP-CS-Fixer! We are a team of software engineering researchers and we have worked on finding out how to best document rules from tools such as PHP-CS-FIXER (that was included in our population of linters :-D). Our results are available on our replication website : https://icpc2024-asats.github.io/ (including full paper and data). We hope that you'll find this useful (and don't hesitate to provide us with any feedback)!

↑ 1

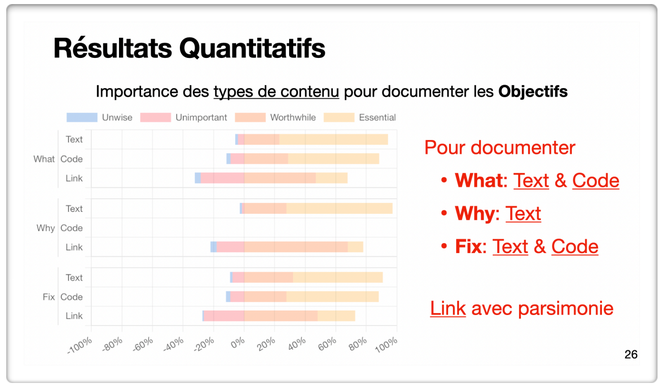
Inférer Automatiquement une Correction

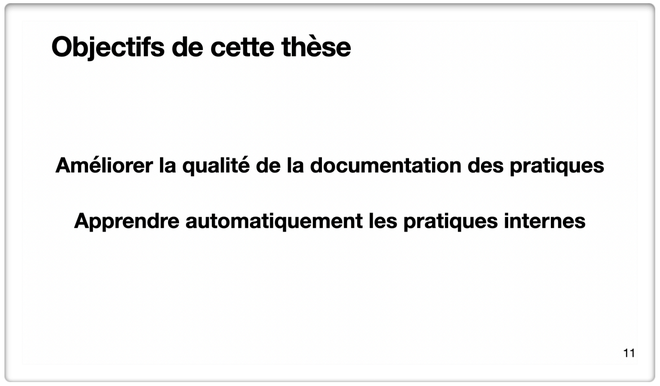
```
fetch( input: '/data/results.json');
```

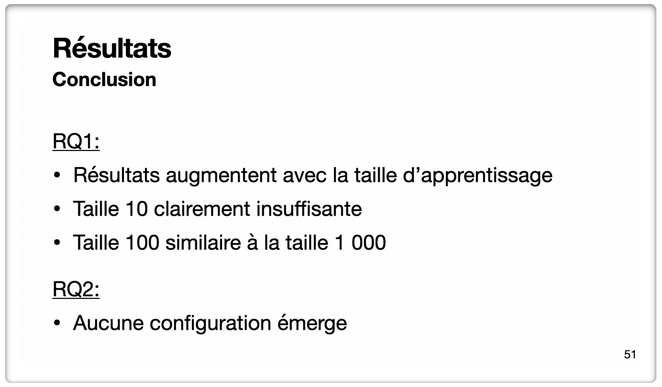
```
fetch( input: `/data/results.json?t=${ Date.now() }`);
```

Maintenir à Jour la Base de Connaissances

Name	Categories	Status	Detection	Added by	Created on
Log backend operations to record runtime behavior	log	Validated	Configured	Cédric Tey...	3/6/2024
Use URLSearchParams for HTTP Requests instead of harcoding queries in URLs		Validated	To configure	Cédric Tey...	2/19/2024
Key should be an unique, immutable property	React	Validated	To configure	Quentin	2/5/2024
Theming angular material	MaterialUI angularJS	Validated	To configure	Malo	1/29/2024







Les Pratiques de Code : De la Documentation à la Détection

C. Latappy, Q. Perez, T. Degueule, J.-R. Falleri, C. Urtado, S. Vauttier, X. Blanc, et C. Teyton,
« MLinter: Learning Coding Practices from Examples—Dream or Reality? »
in *2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), mars 2023*
doi: 10.1109/SANER56733.2023.00092.

C. Latappy, T. Degueule, J.-R. Falleri, R. Robbes, X. Blanc, et C. Teyton,
« What the Fix? A Study of ASAT Rules Documentation »
in *Proceedings of the 32nd IEEE/ACM International Conference on Program Comprehension, ICPC 2024*
doi: 10.1145/3643916.3644404.

université de BORDEAUX LaBRI packmind