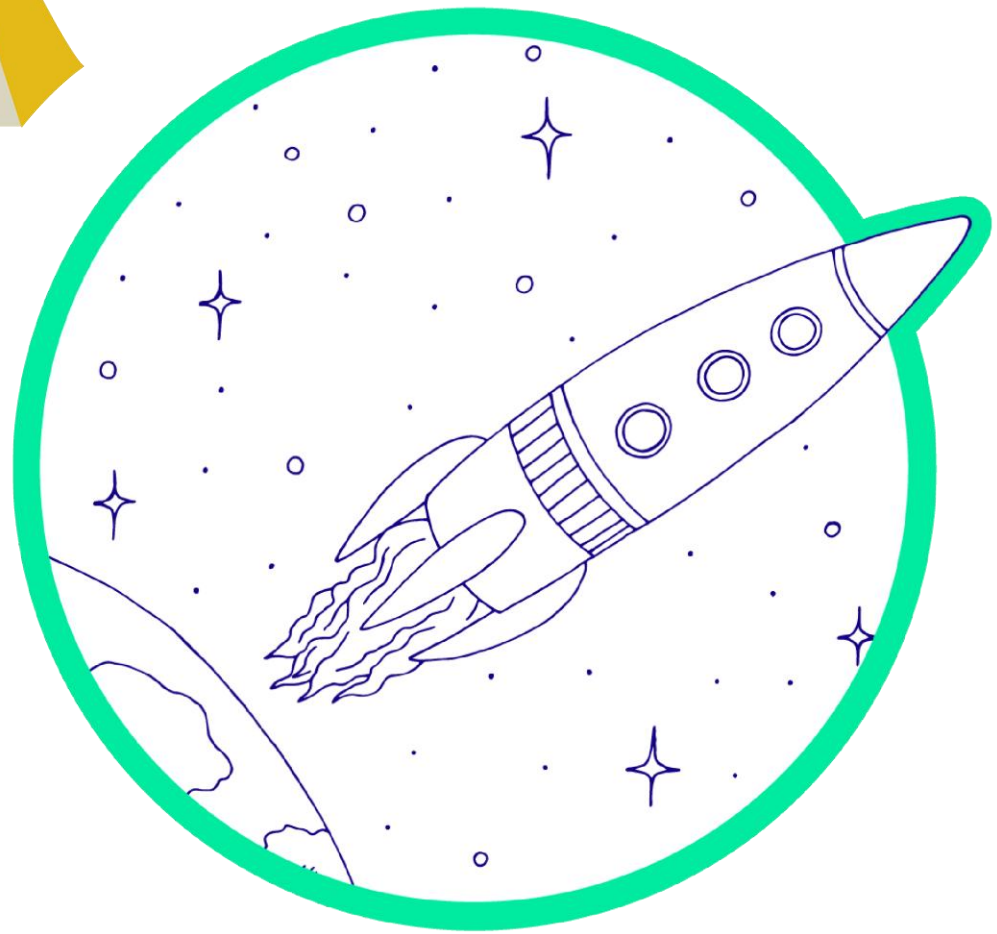
CRITEO <R&D>

An expert community
where every voice matters

+



Microservices, macrogalères?

Repensez vos tests pour survivre!

Un Grand MERCI à nos sponsors 2026

CGI

open

KAIZEN SOLUTIONS

aws

CRITEO

Moody's

clever cloud

AVISTO

KLS GROUP

kelkoogroup

VISEO
POSITIVE DIGITAL MAKERS

alma

sopra steria

PeriScop

enalean
software engineering is now

HACK YOUR JOB

PingID

zenika

Arnaud Becquet

Señor Software Engineer

Benjamin Baumann

Señor Engineering Manager

@zentiltoutou

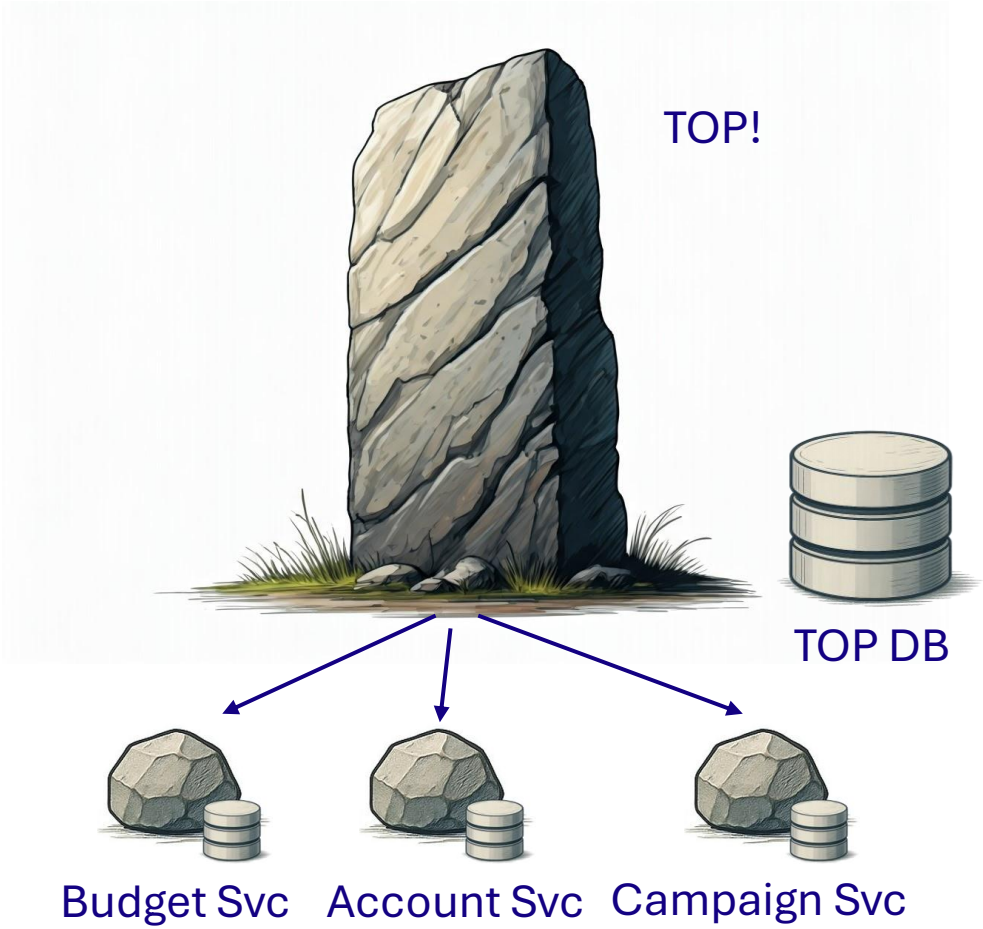
CRITEO <R&D>

Agenda

1. Avant les microservices...
2. Stratégie microservice
3. Les macrogalères
4. En direct du terrain : feedback sur les techniques de test

Avant les migroservices

Retour en 2019



- Monolithe + quelques services (CRUD)
- Logique dans le monolithe
- Release hebdomadaire

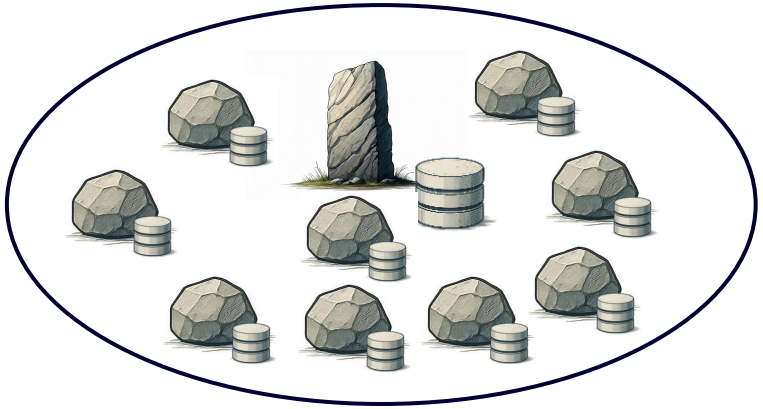
Avant les migroservices

Comment on testait ça?

E2E tests (Selenium)

Unit Unit • • • Unit Unit

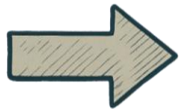
- Build : Tests unitaires (UI: jasmine, backend : NUnit)
- CI : E2E tests dans Sandbox



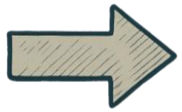
- Validation manuelle en preprod

Avant les migroservices 

- 3 sandboxes
- De plus en plus de services
- Logique distribuée
- Plusieurs ~~monolithes~~ applications
- Rythme de release challengeant
- Explosion des coûts de sandbox

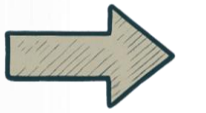


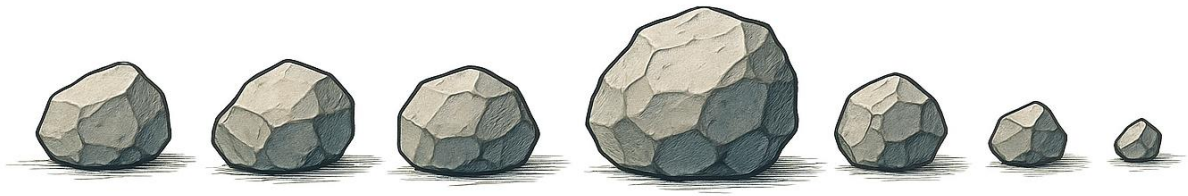
On change notre archi système



On doit adapter notre stratégie de test

CRITEO <R&D>





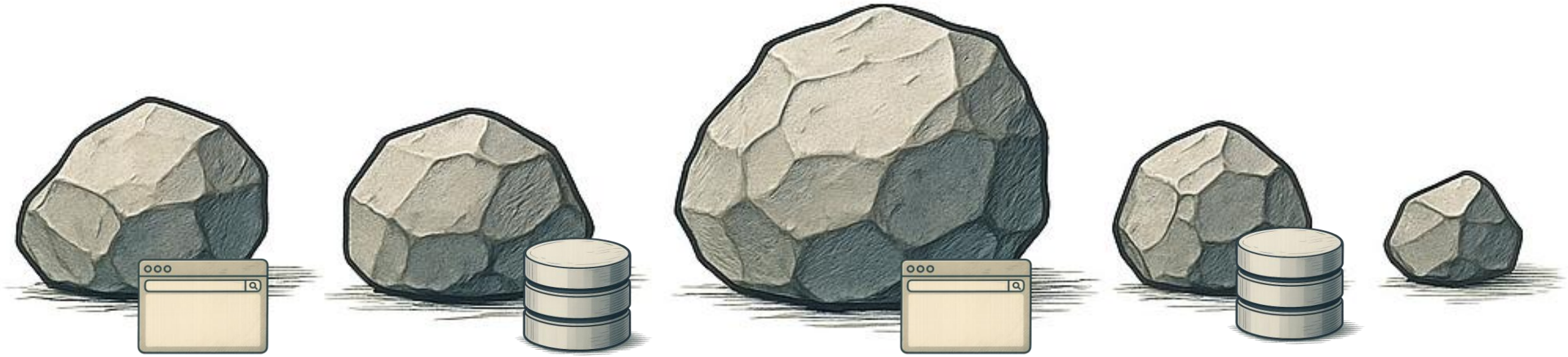
E2E tests (Selenium 🦖)

Unit Unit • • • Unit Unit

?

CRITEO <R&D>

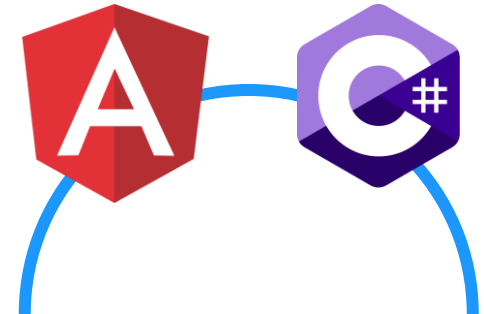
Notre stack (enfin celle dont on parle ici)



REST APIs par domaine associées à une DB

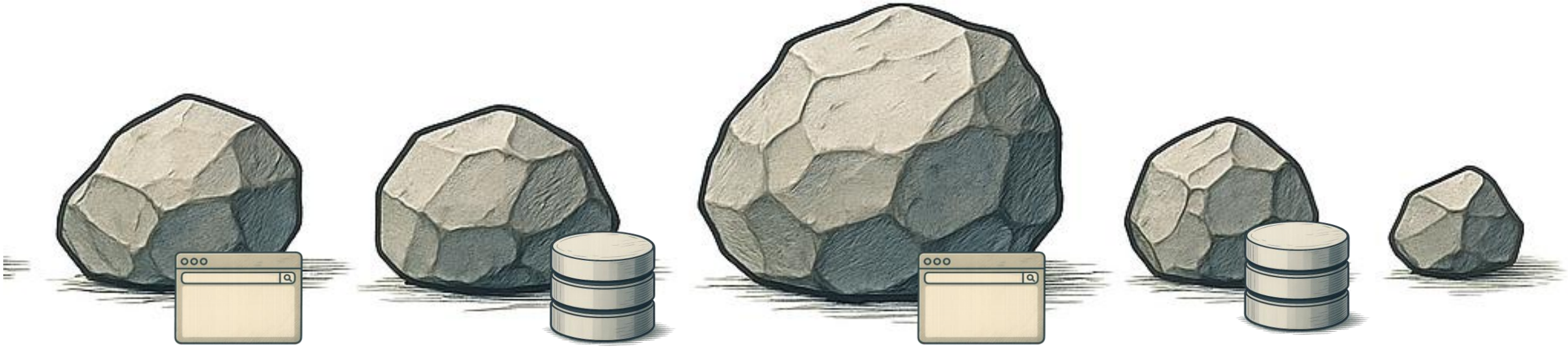
&

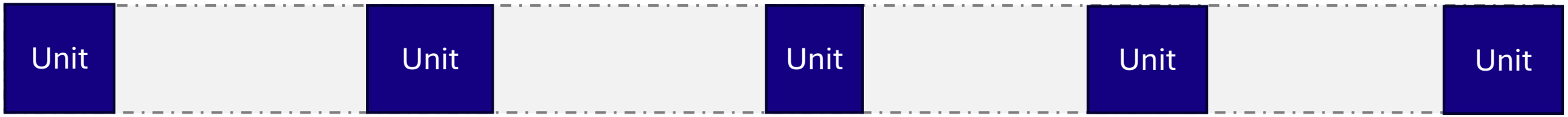
Applications frontend associées à un backend/gateway



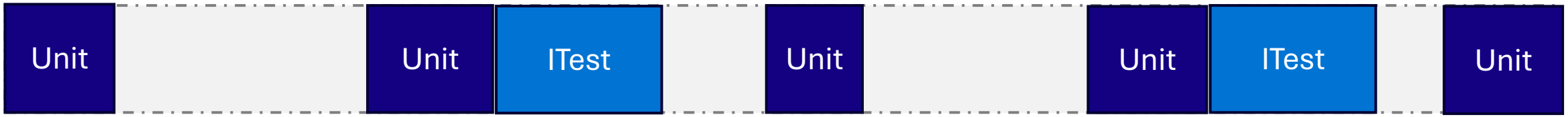
CRITEO <R&D>

Test unitaire, la base



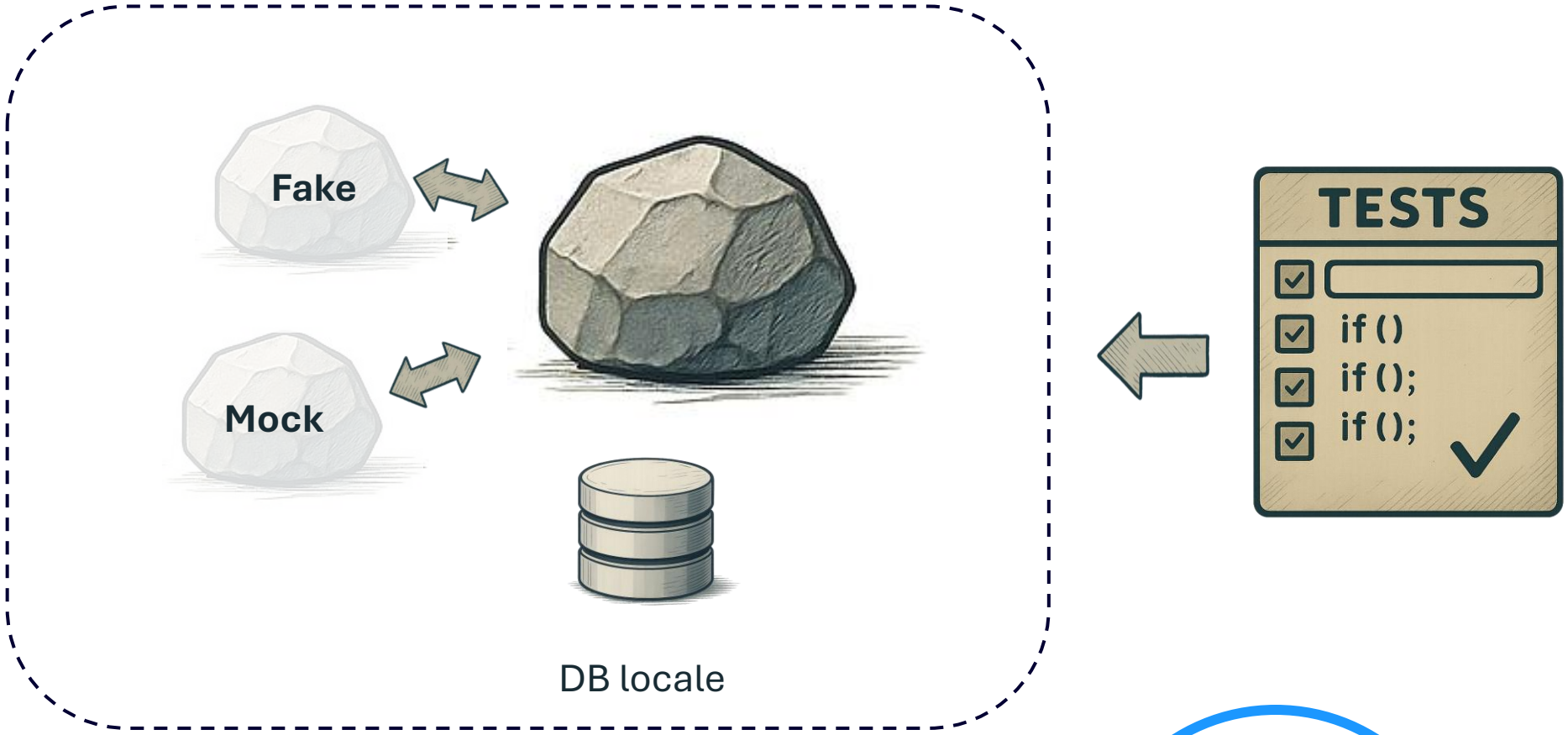


ITests aka Tests en isolation



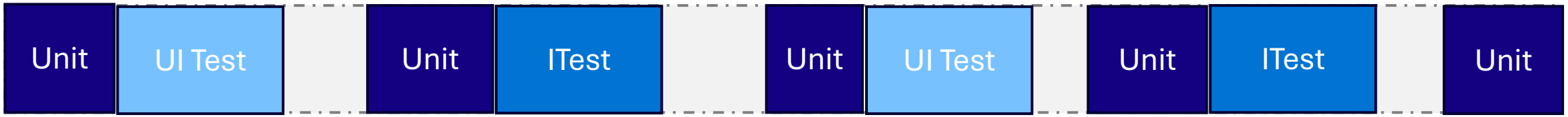
CRITEO <R&D>

Tests en isolation

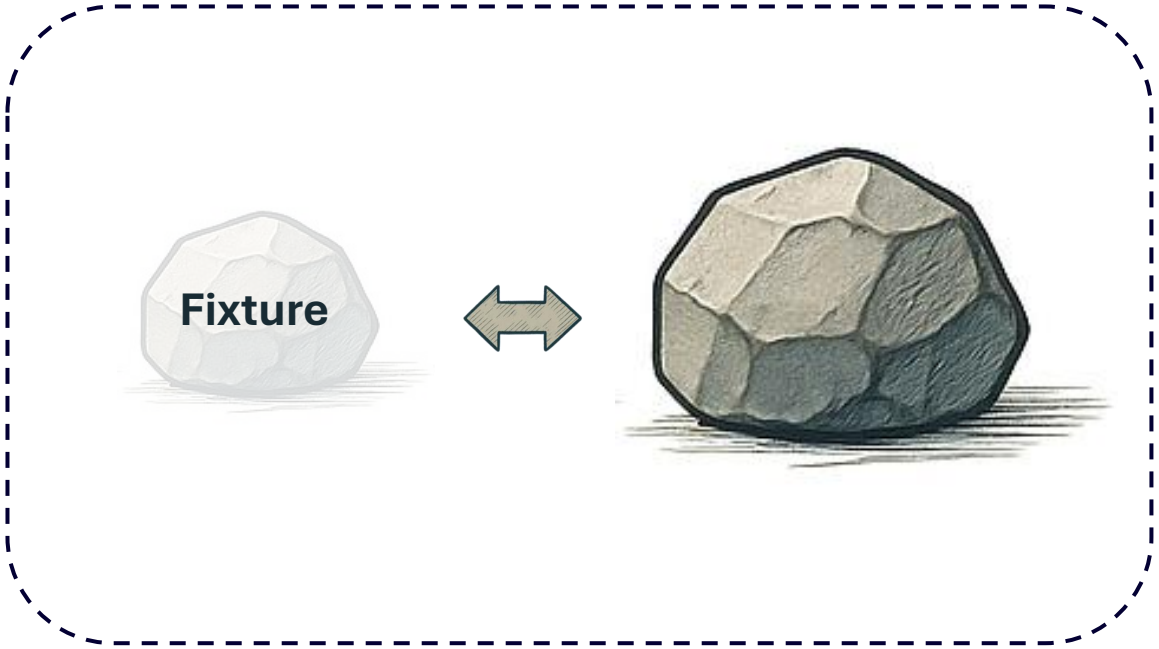


Application en isolation

Tests UI



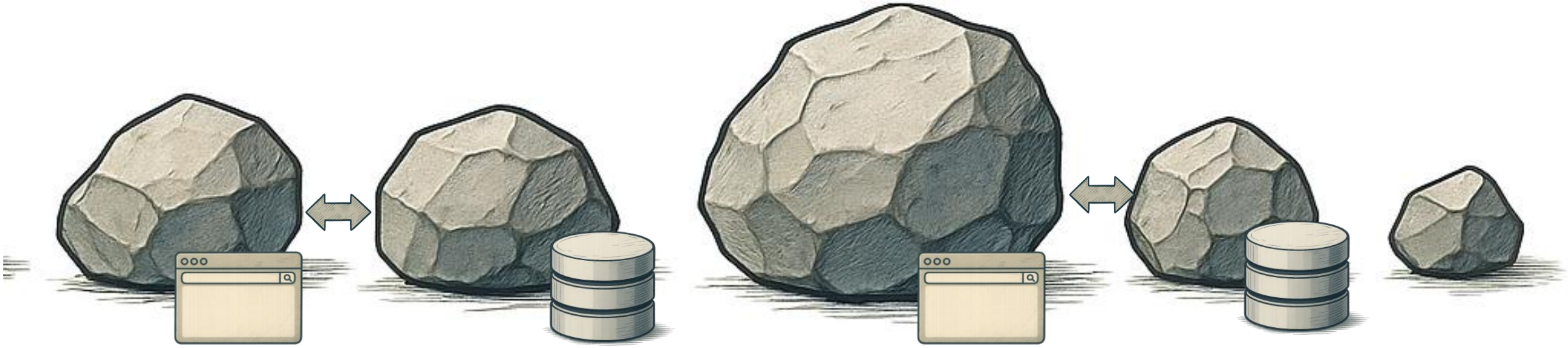
Tests UI

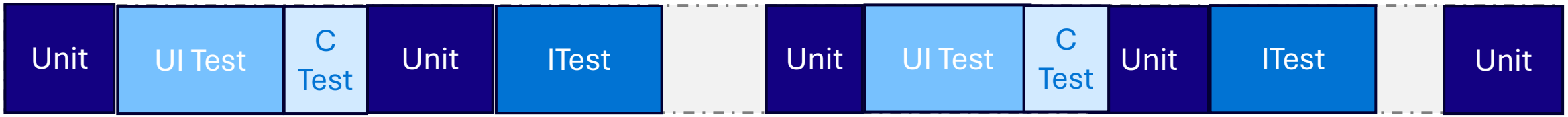


cypress

Playwright

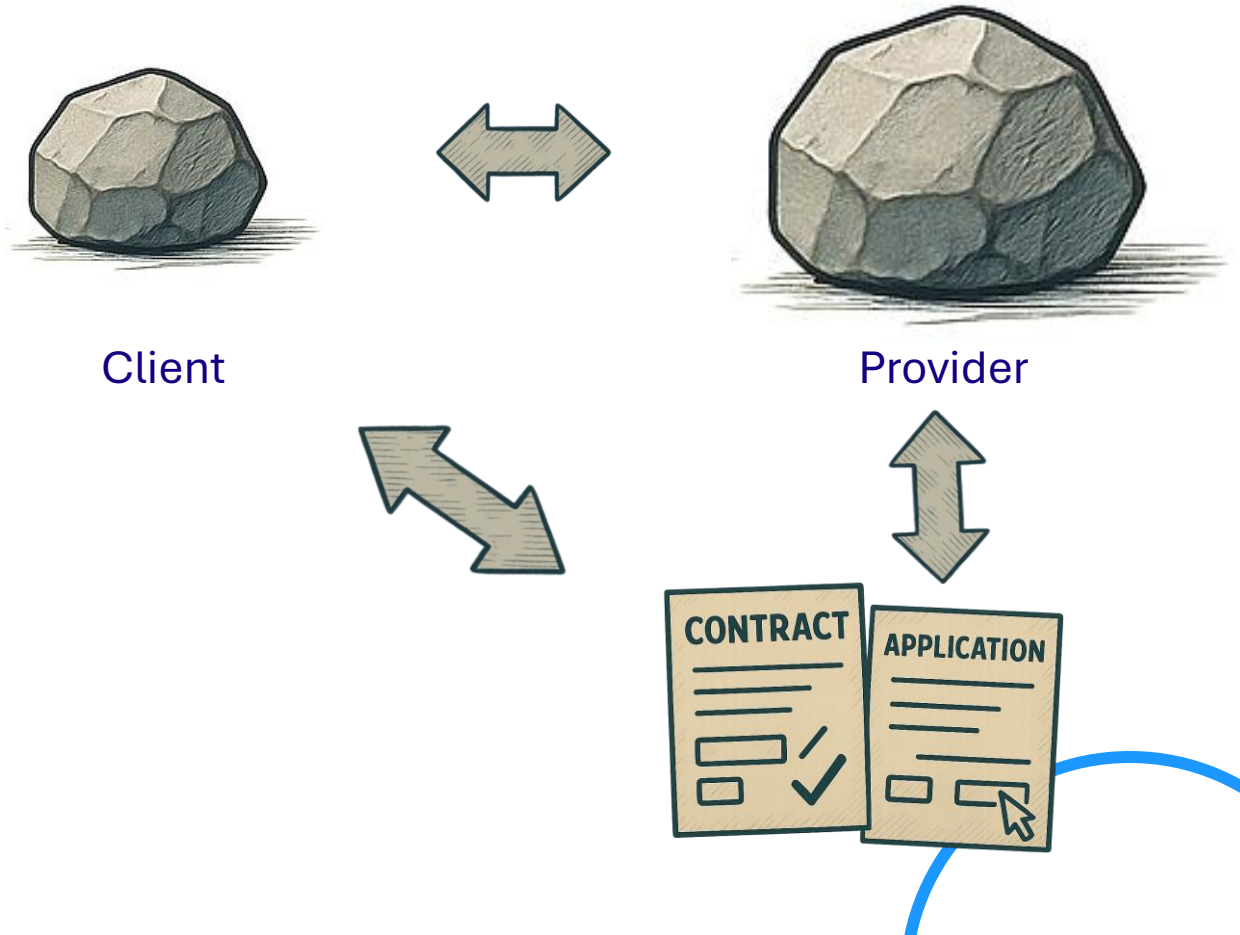
Tests de contrat



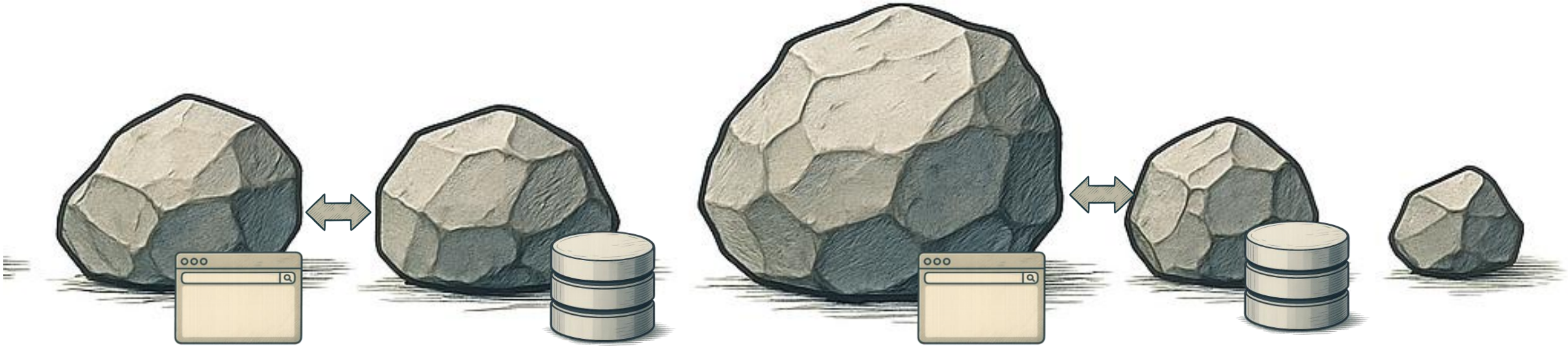


CRITEO <R&D>

Tests de contract orienté Provider

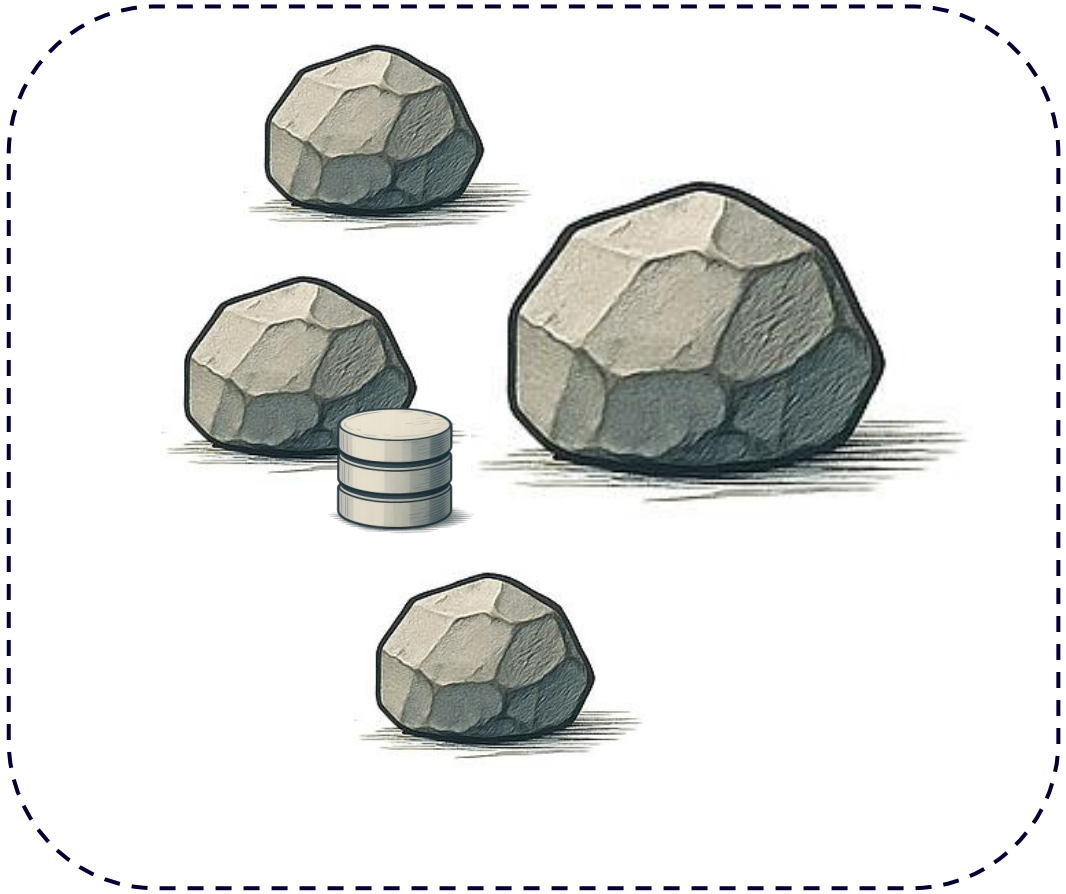


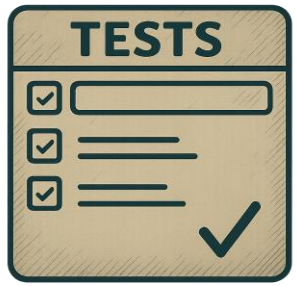
E2E test en préproduction



CRITEO <R&D>

E2E test en préproduction



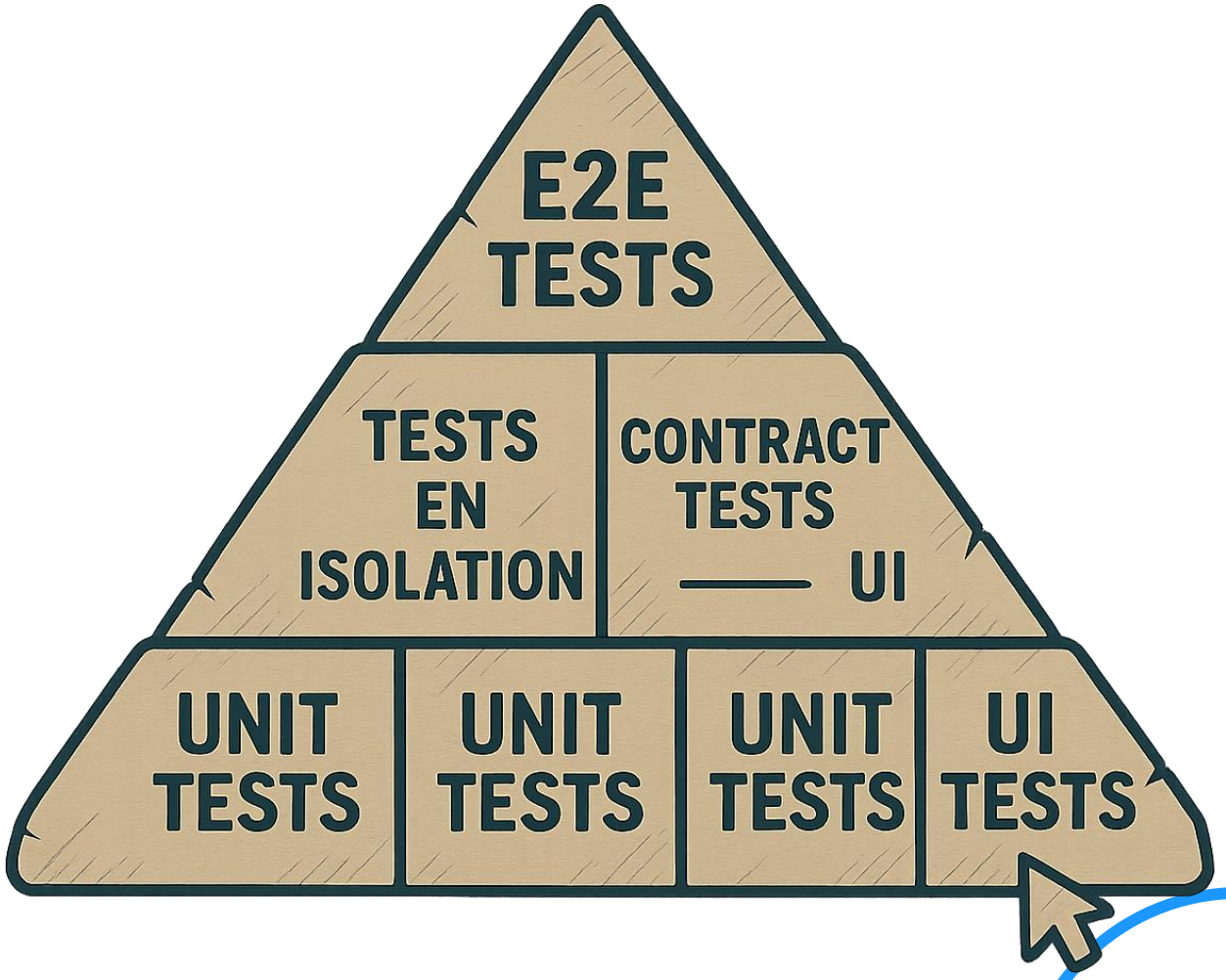


POSTMAN

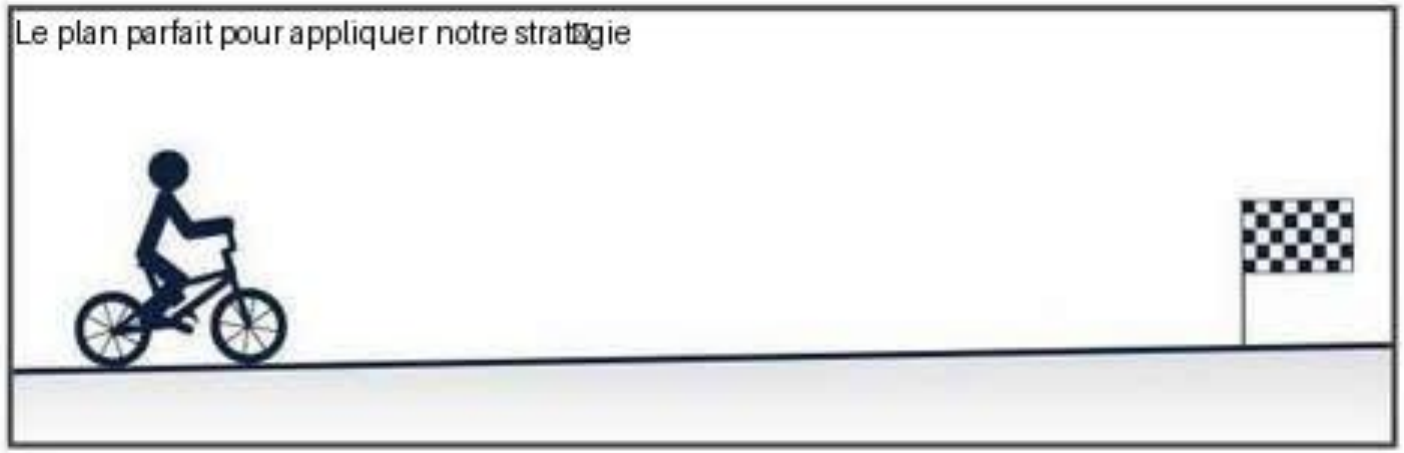
cypress

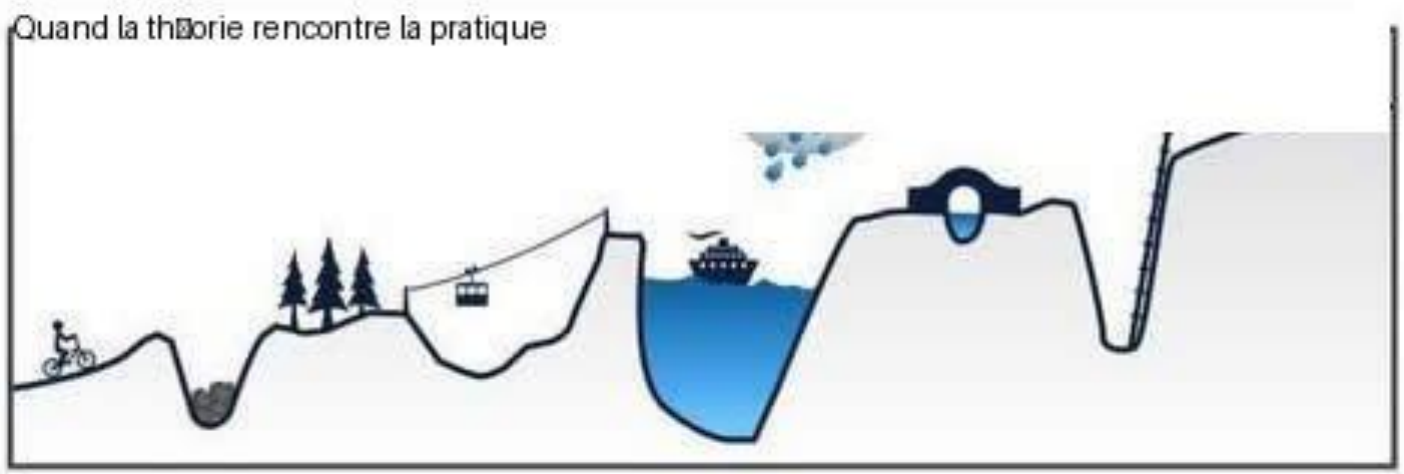
CRITEO <R&D>

Pyramide


E2E
TESTS
TESTS
EN
ISOLATION
CONTRACT
TESTS
UI
UNIT
TESTS
UNIT
TESTS
UNIT
TESTS
UI
TESTS

Les macrogalères





Tests en isolation

Nos problèmes

- Setup nécessitant une bonne connaissance de l'injection et test double 🚧
- Compatibilité avec les OS des workstation
- Lent (démarrer et restaurer les DBs 🐌)

Nos solutions

- Test double plus obligatoire 🔗
- Docker pour tous 🐳
- Dacpac 🚀

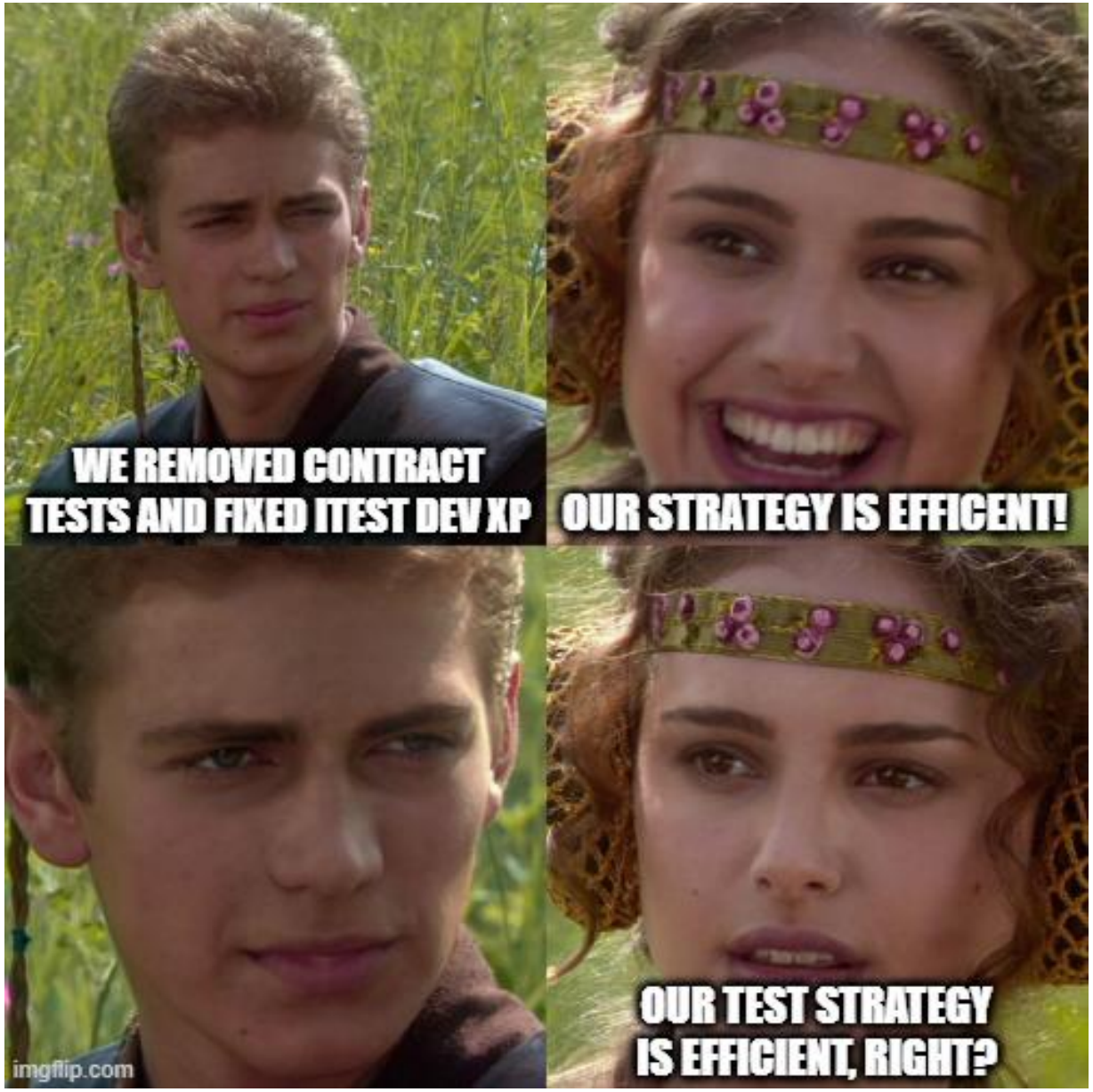
Tests de contrat côté Provider

Nos problèmes

- Dépendance forte entre le provider (test double) et le client 👉

Nos solutions

- Ne pas en faire: client auto-générés couvre l'intégration entre les applications 🔗



- Les refactoring techniques cassaient les tests
- Obligés de rajouter des tests "haut niveau" pour refactorer avec confiance.
- Fatigue des devs

CRITEO <R&D>

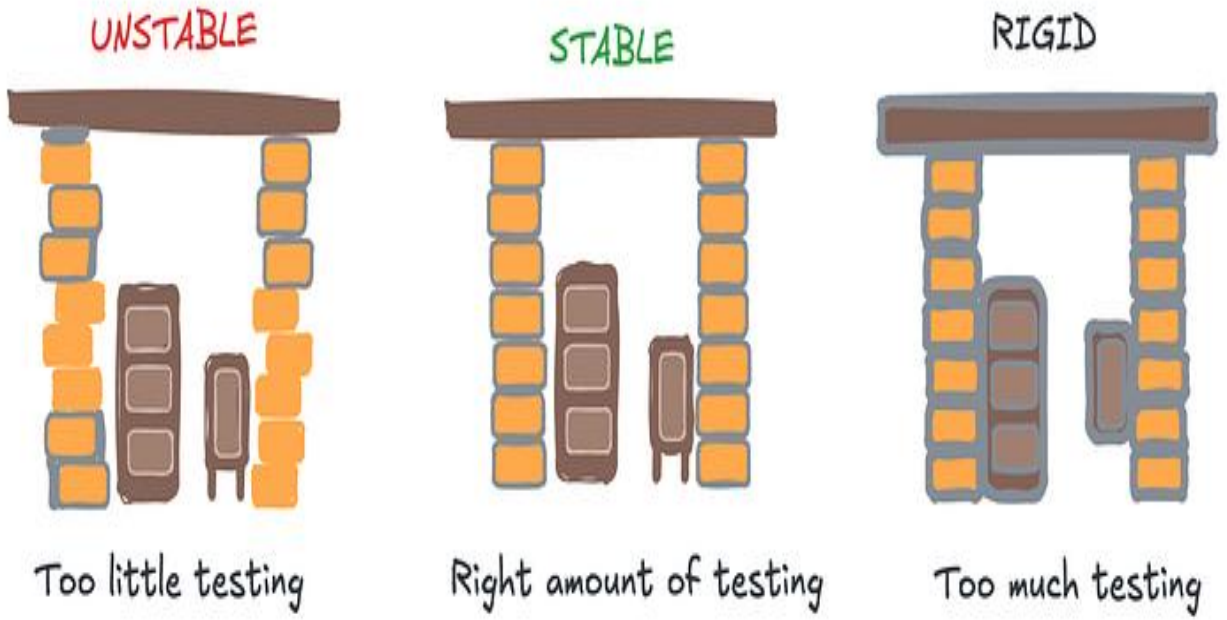
Prenons du recul

CRITEO <R&D>

Test = Ciment

- Trop tester c'est cimenter les mauvaises briques

- Symptomes
  - Changer les tests durant un refacto
  - Plus de tests "techniques" que "métier"
  - Impression de tester la même chose N fois
  - Mock à tous les étages



On n’a pas assez réfléchi...

Me writing code

inversion of control, rule engine, DRY, SOLID, Code quality...

Me writing tests

CopyPasta

On écrit souvent du bon code.
On écrit rarement des bons tests.
C'est souvent parce qu'on ne sait pas ce qu'on doit tester

CRITEO <R&D>

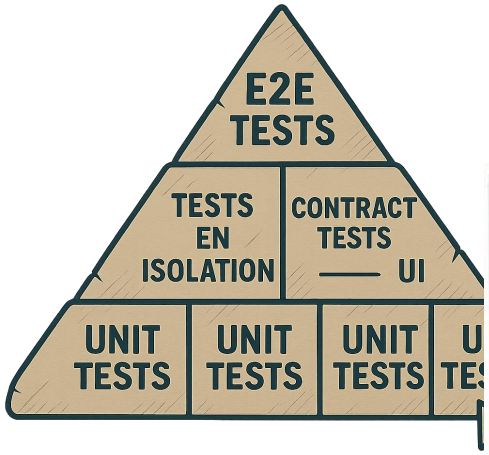
L’autopilote a créé un monstre

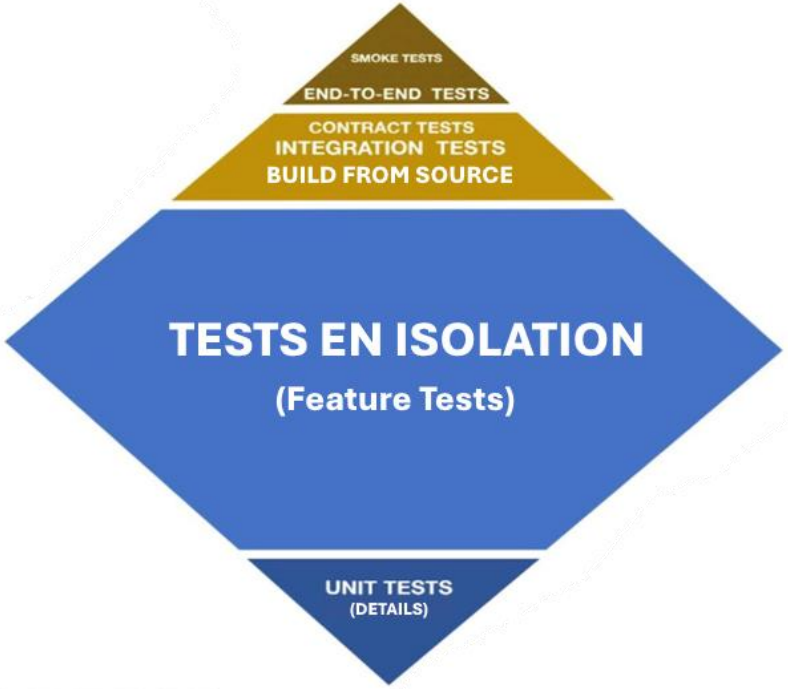
CRITEO <R&D>

Une bien meilleure stratégie

- Tester la feature pas le code
- Blackbox
- Cimenter ce qui a de la valeur
- Débrancher l'autopilote

Repenser notre pyramide... ou notre Unit

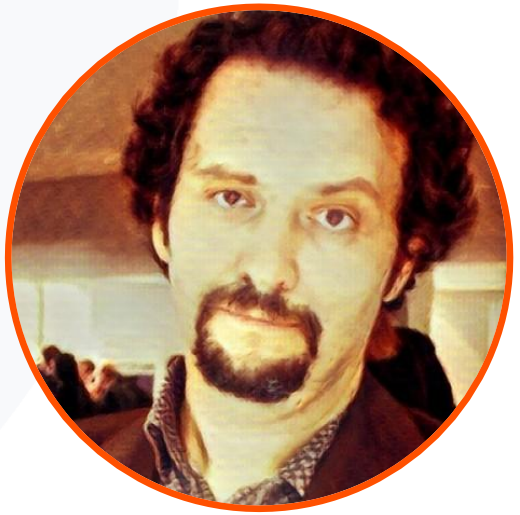




CRITEO <R&D>

“You write a new test when you need to fulfill a requirement. **You do not write a test when you need to code a new class or a new method**.

It will lead your codebase to excessive coupling between tests and implementation details and your tests will break whenever refactoring occurs.”

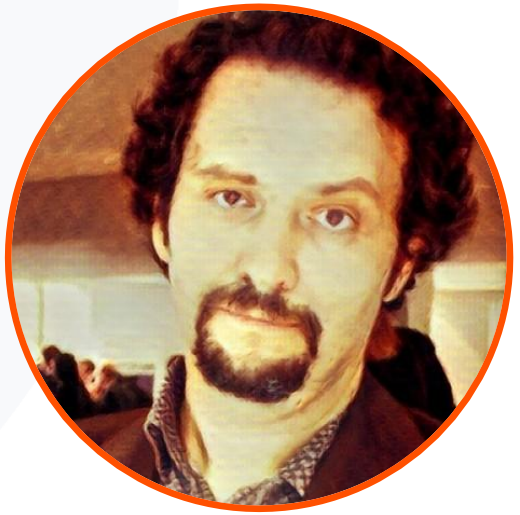


Cyrille Dupuydauby – 2018
https://medium.com/@Cyrdup/unit-testing-youre-doing-it-wrong-407a07692989

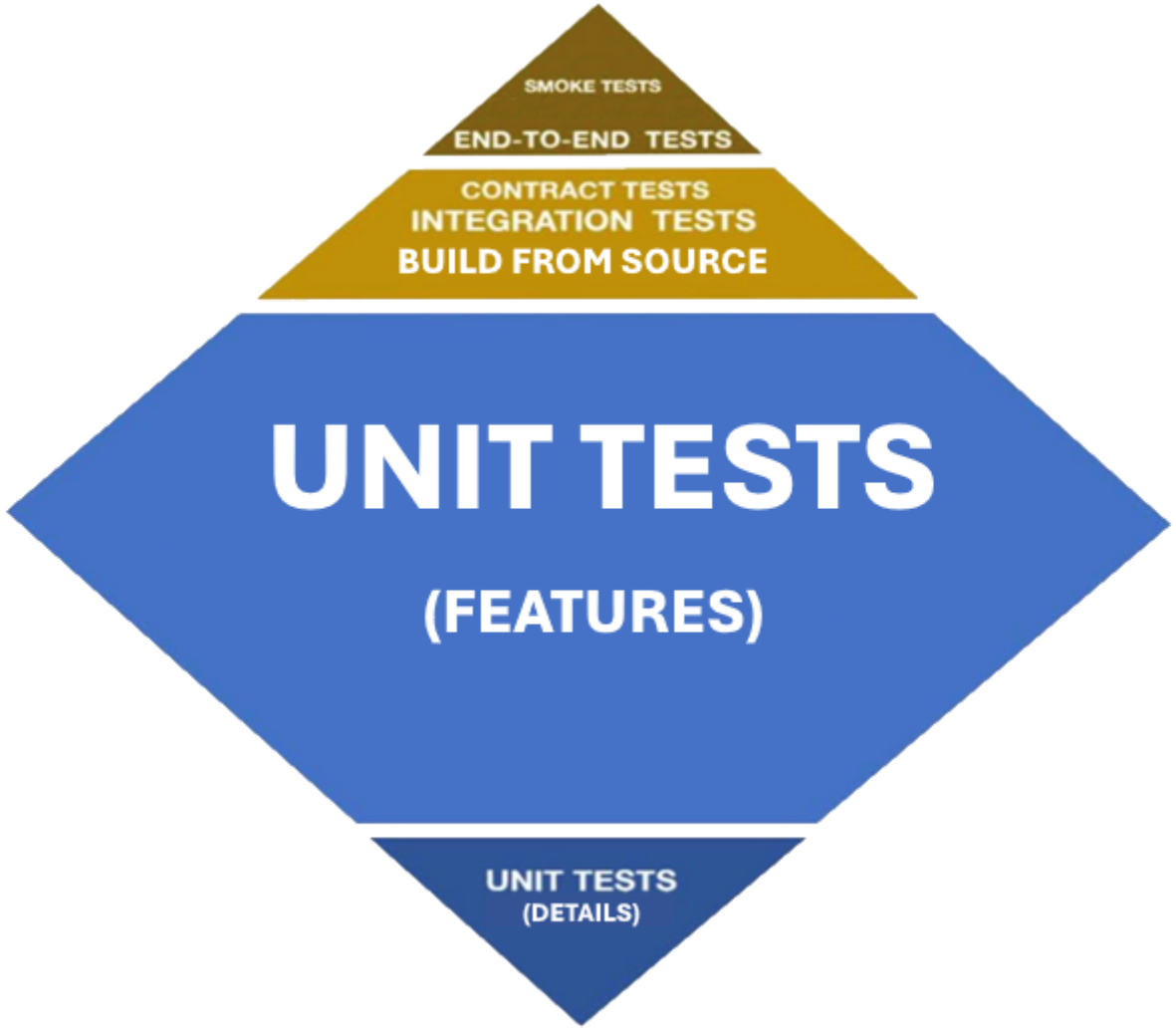
Often teams are using mocks to test classes in isolation, mocking every dependencies. You don't need to do that! Remember, **‘unit’** in unit testing is to be understood **as a module or a component**, not a class.

Whenever you decide to introduce a mock, you enforce a contract that makes refactoring more difficult. Mocks are here to help you get **rid of slow or unstable dependencies**, such as a remote services, or some persistent storage.

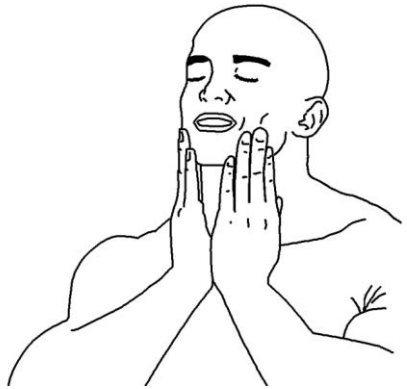
Cyrille Dupuydauby – 2018
https://medium.com/@Cyrdup/unit-testing-youre-doing-it-wrong-407a07692989



Notre stratégie

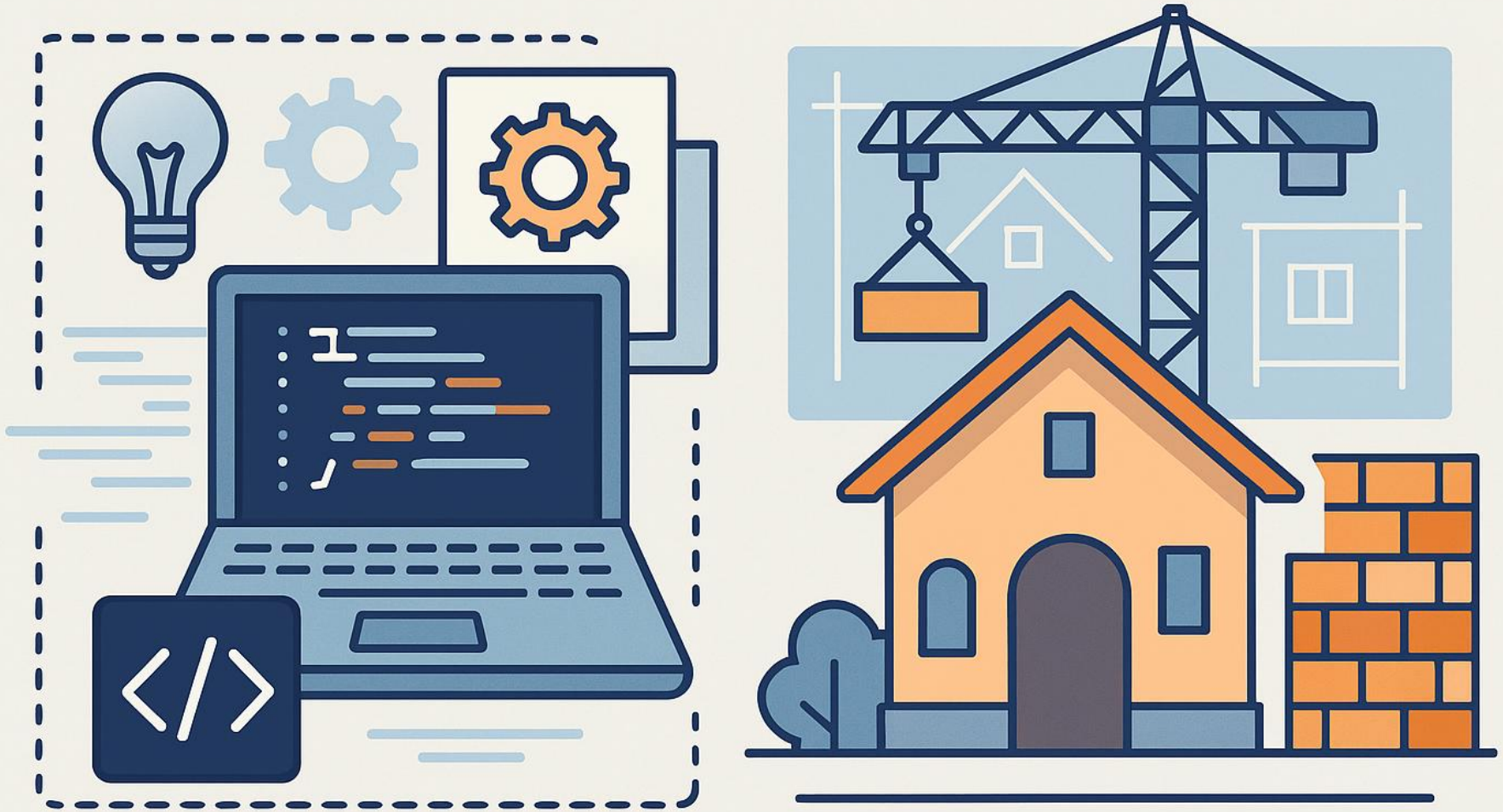


- Unit = Feature
- Rapide, orienté métier
- Cimente ce qui est important
- Bonne couverture
- Release quotidienne
- Confiance et robustesse



CRITEO <R&D>

MÉTHODES ET PATTERNS



Sociable UTest


```
class ProductService {
  PricingEngine engine;

  public ProductService(PricingEngine engine) {
    this.engine = engine;
  }

  public double getPriceFor(Product product) {
    return engine.calculatePrice(product.getCost());
  }
}

class PricingEngine {
  double markup;

  public PricingEngine(double markup) {
    this.markup = markup;
  }

  public double calculatePrice(double cost) {
    return cost * markup;
  }
}
```

```
// Solitary Unit Test

@Test
public void shouldGetPrice() {
  PricingEngine engine = mock(PricingEngine.class);
  Product product = new Product(10);

  when(engine.calculatePrice(10)).thenReturn(13);

  ProductService productService = new ProductService(engine);
  double price = productService.getPriceFor(product);

  assertThat(price).isCloseTo(13.0, within(0.1));
  verify(engine, times(1)).calculatePrice(10)
}
```

```
// Sociable Unit Test

@Test
public void shouldGetPrice() {
  Product product = new Product(10);

  ProductService productService = new ProductService(new PricingEngine(1.3));

  double price = productService.getPriceFor(product);

  assertThat(price)
   .isCloseTo(13.0, within(0.1));
}
```

Source : https://dev.to/dylanwatsonsoftware/socialise-your-unit-tests-5da0

CRITEO <R&D>

System Under Test Builder

Des tests oui mais des tests lisibles!

```
[Test]
public async Task Validate_AdSets_Are_Eligible_When_Resetting_CategoryBidAmounts()
{
    // Arrange
    var archivedAdSetId = 46657; var adaptiveBusinessModelAdSetId = 496;
    var invalidNullBidAmountAdSetId = 666; var nonexistentAdSetId = int.MaxValue;

    var sut = CategoryBidAmountResetCommandBuilder
        .Create()
        .WithArchivedAdSet(archivedAdSetId)
        .WithIncompatibleAdSet(adaptiveBusinessModelAdSetId)
        .WithInvalidNullBidAmountAdSet(invalidNullBidAmountAdSetId)
        .Build(out _);


    // Act
    var result = await sut.ResetCategoryBidAmount(...)
    // Assert
    Assert.That(...)
}
```

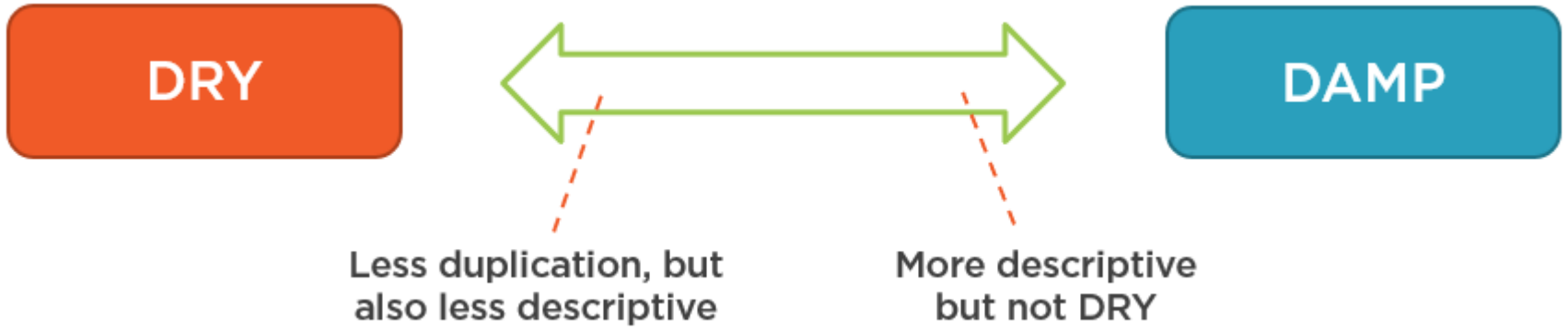
- Construction de la unit à part (SUTBuilder)
- Fluent
- ✅ Tests sont lisibles
- ✅ Quand on refacto, la feature ne change pas, les tests non plus!
- ✅ Réutilisable (tests sociables...)
- Construire un SUTBuilder nous fait réfléchir au SUT :)
- Pour des tests existants, migrer avec l'aide de l'IA

#protips

CRITEO <R&D>

CRITEO

Les tests doivent être descriptifs et lisibles (DAMP)



Property Based Testing (PBT)

⭐⭐ Du potentiel mais peu de use cases en pratique.

Des maths???

```
[FsCheck.NUnit.Property()]
public Property TryParse_Sucessfully(int adSetId, int categoryHashcode, int categoryColumnId)
{
    string ToModelAndBack(string key)
    {
        DisplayMultiplierCategoryKeyMapper.TryParse(key, out var displayMultiplierKey);
        return DisplayMultiplierCategoryKeyMapper.GetDisplayMultiplierIdFrom(displayMultiplierKey);
    }
    var validKey = $"{adSetId}|{categoryHashcode}|{categoryColumnId}";
    return (validKey == ToModelAndBack(validKey)).ToProperty();
}
```

- ❗ PBT s'assure que les **propriétés** du système sont vraies pour toute entrée.
- Generators/Shrinkers (emoji, maxvalue...)

#protips

- ✅ Remplace les TestCases
- ✅ Super pour les mappers
- ⚠️ Dur de trouver les propriétés

CRITEO <R&D>

Mutation Testing

Sympa quand on a du temps et qu'il fait trop moche pour aller en montagne

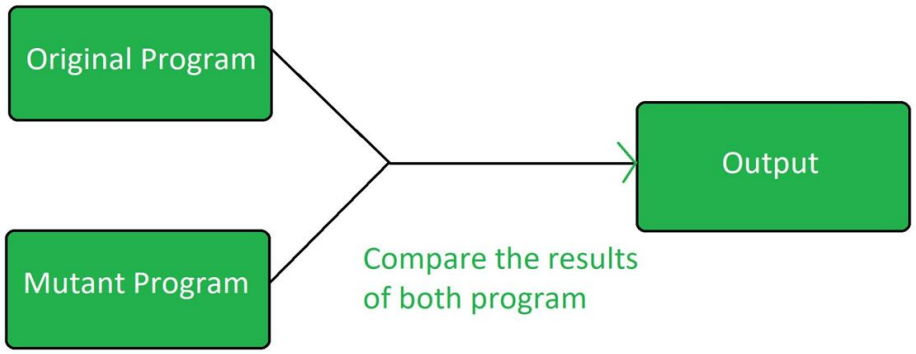
Tester ses tests

```
Initial Code:

if(a < b)
 c = 10;
else
 c = 20;

Changed Code:

if(a > b)
 c = 10;
else
 c = 20;
```



- Fun
- Valeur ajoutée rapide (quick fix)
- Lent à exécuter
- "One off" / une fois par an

CRITEO <R&D>

Approval Testing

🛟 Rarement utile mais peut sauver la vie

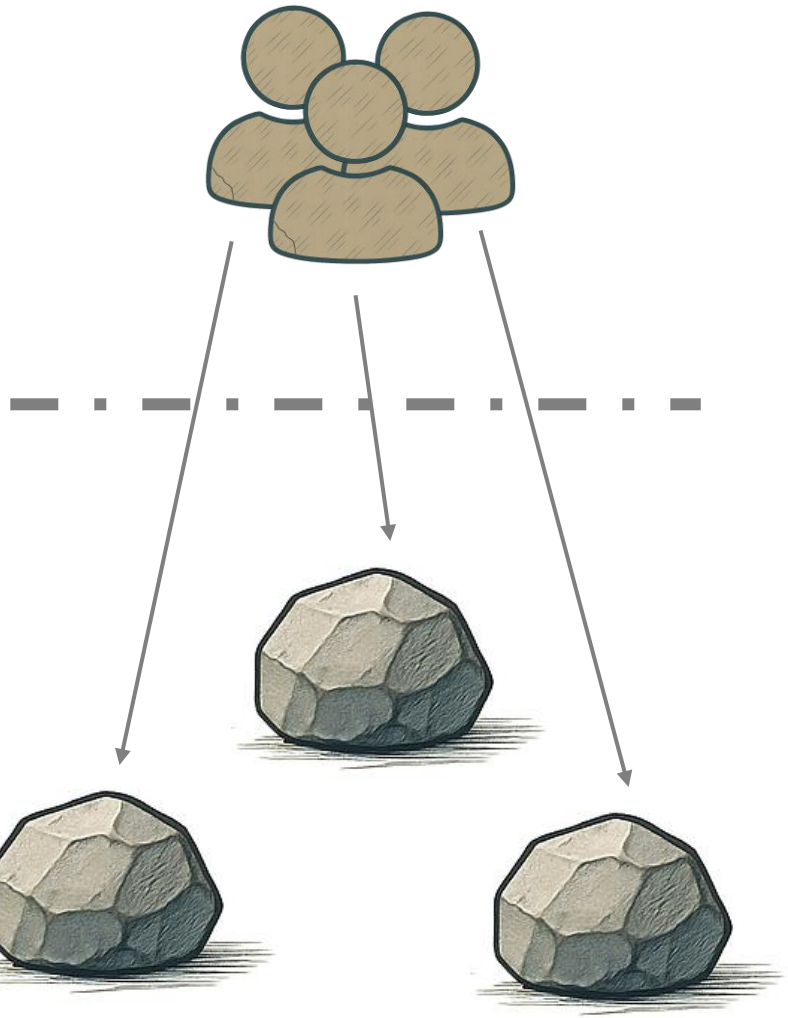
Quand il n'y a plus de spec, l'existant c'est la spéc

A utiliser quand

- Code pas testé
- Code compliqué à comprendre
- La deadline est la semaine prochaine

1. Génère un snapshot de input/ouput
2. Test Coverage pour vérifier
3. Changer le code
4. Supprimer les tests !

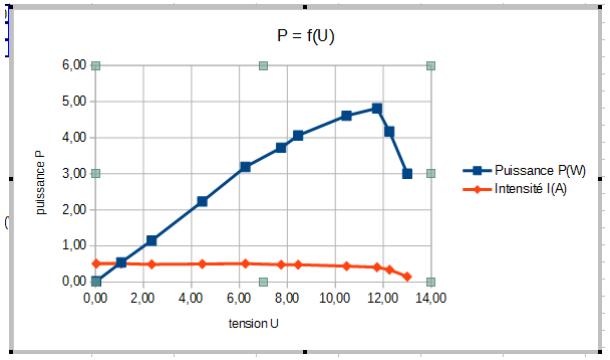
Traffic Replay



Version à tester

Version de prod

Essentiel pour tester des modèles non déterministes mais nécessite un fort trafic



Master Arbitrage MOAB #104343: SUCCESS

MOAB	Log	Start time	
104343	/	15/01/2026 08:15:43	15/
104342	/	15/01/2026 00:40:52	15/
104341	/	15/01/2026 00:13:48	15/
104340	/	14/01/2026 22:59:01	14/
104339	/	14/01/2026 22:28:22	14/
104338	/	14/01/2026 22:06:13	14/

CRITEO <R&D>

Behavior-Driven Development (BDD)

Quoi

Rédiger des **tests** sous forme de scénarios en **langage clair** pour décrire les comportements attendus

Quand

Quand la collaboration entre les parties prenantes **techniques** et **non techniques** est nécessaire

Pourquoi

- Améliore la communication et la compréhension
- Aligne les tests sur les besoins de l'entreprise


```
Scenario: Monitor is created with no line items. As a result - 0 count is emitted.
      Given the monitor "retailmedia.kobalos.line_items_no_budgets"
       When LineItemNoBudget is created
       Then 0 count is emitted for the metric "retailmedia.kobalos.line_items_no_budgets"
```

1. Tester les features et leur intégration pas le code
2. Tester c’est cimenter
3. Débrancher l'autopilote, challenger ses pratiques
4. Penser aux patterns de tests

LE TEST C’EST DU CODE BORDEL!!!

Resources

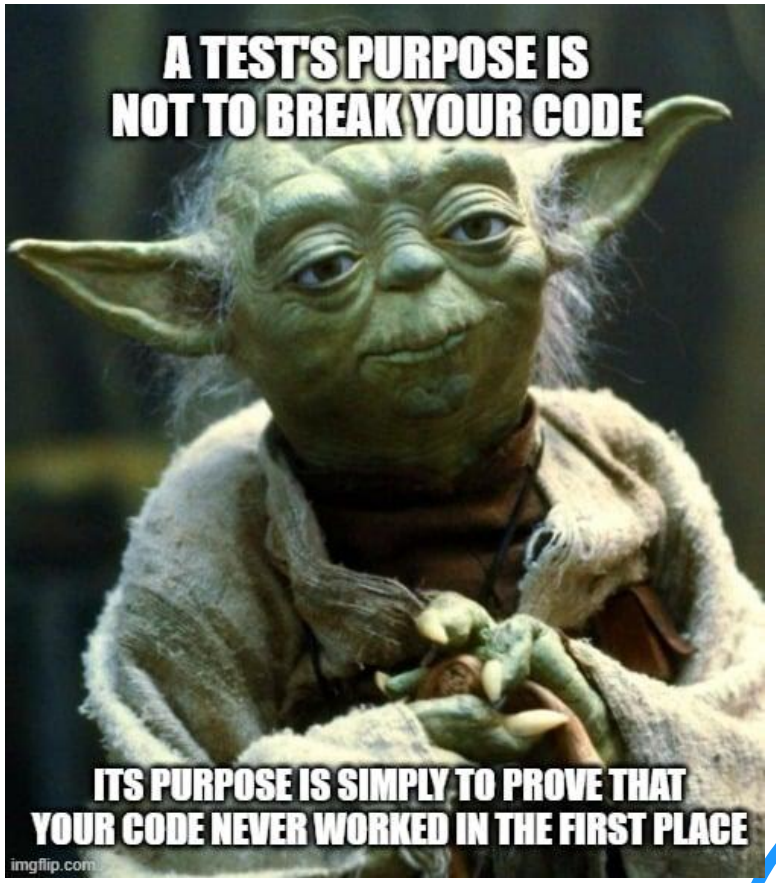
- https://martinfowler.com/articles/2021-test-shapes.html
- The cement of software : deep dive into testing, Criteo Blog, Paola Valdivia & Benjamin Baumann
- Unit testing, you're doing it wrong par Cyrille Dupuydauby
- Solitary vs Sociable UTest par Martin Fowler
- On the diverse and fantastic shapes of testing par Martin Fowler
- Canon TDD par Kent Beck
- Socialise your Unit Test par Dylan Watson

CRITEO <R&D>

An expert community
where every voice matters

Merci!

Feedback !



On recrute !

CRITEO <R&D>