

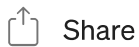
Développement piloté par les tests et comportement(BDD et TDD)



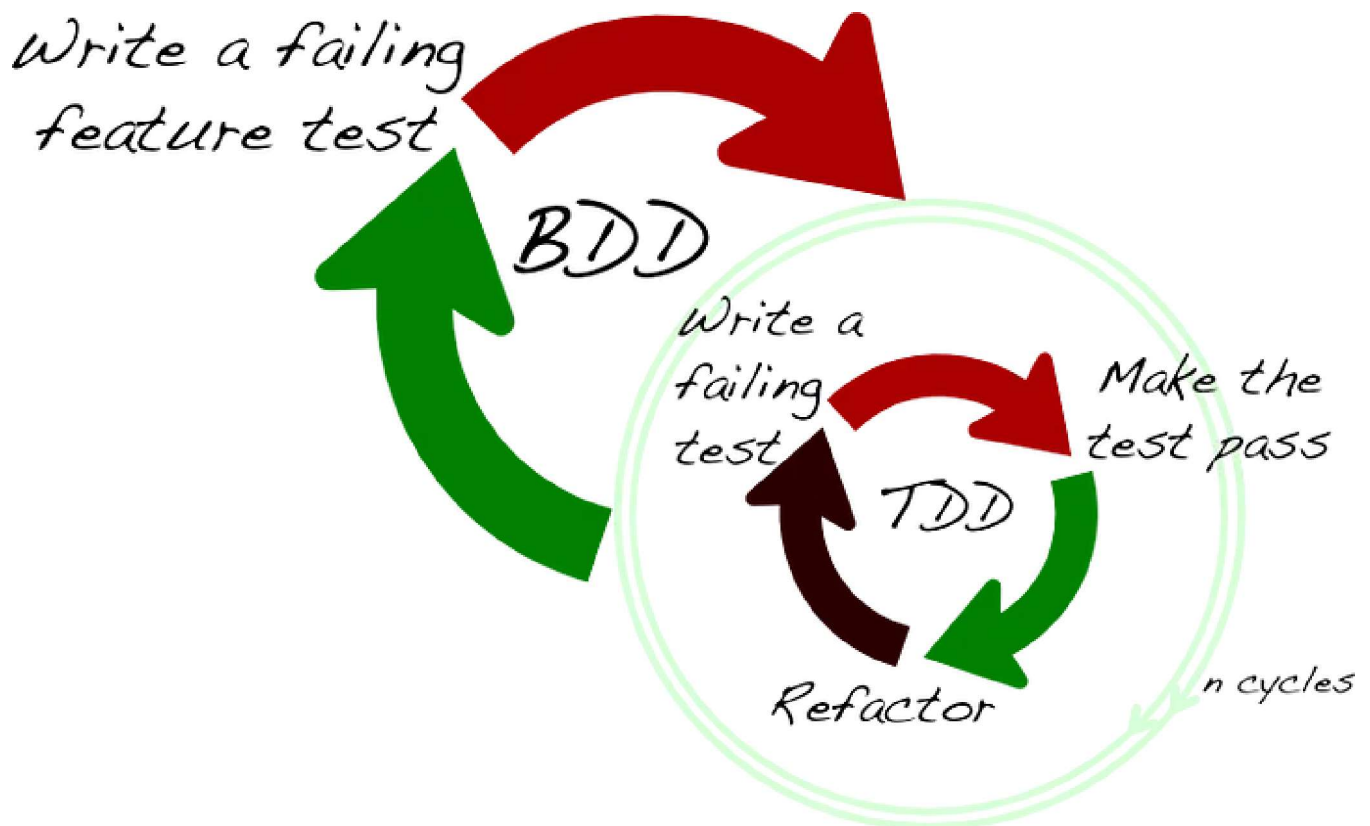
Abdourahmane Sow · Follow

Published in Ouidou

7 min read · Nov 12, 2021



Share



A vant de commencer à entrer de le vif du sujet , je vais vous donner quelques raisons pour lesquelles il est important d'écrire des tests.

1. Rédiger un meilleur code.

2. Débogage plus facile.
3. Minimiser les régressions.
4. Refactoriser en toute confiance.
5. Gagner plus de temps.

Open in app [↗](#)

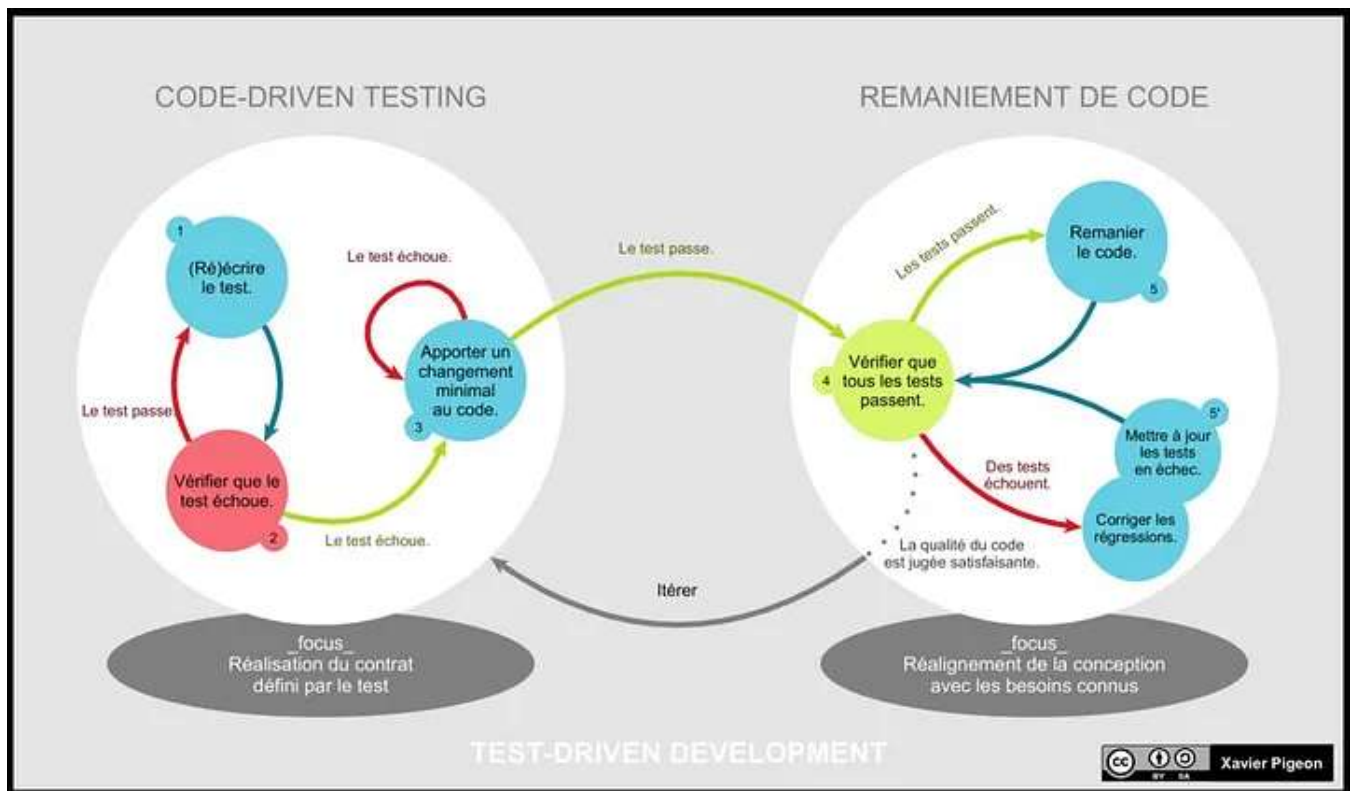
Sign up

Sign In



piloter la conception et le développement de votre application logicielle.

Ce cycle est aussi connu sous le nom de Rouge Vert et Refactor



Cycle Rouge-Vert-Refactor :

L'approche du rouge, vert et refactorisation aide les développeurs à compartimenter leur attention en trois phases :

Rouge: Pensez à ce que vous voulez développer.

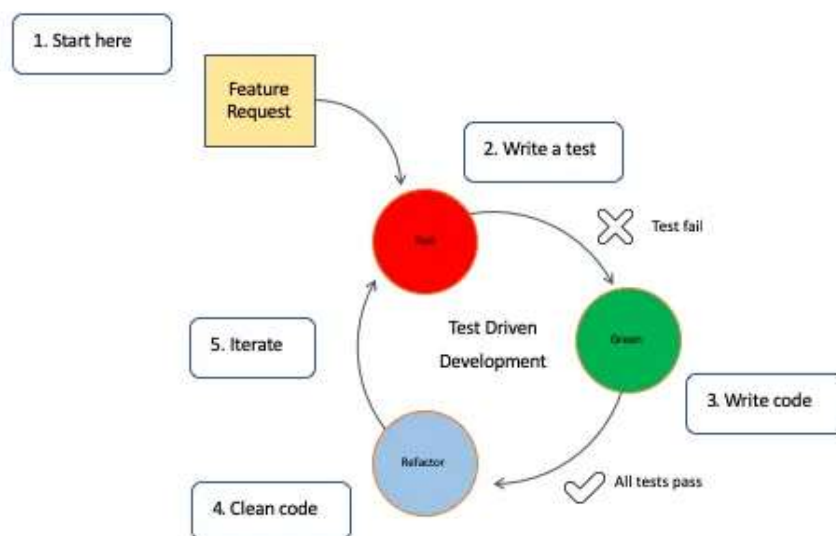
Vert: Réfléchissez à la façon de réussir vos tests.

Refactoriser: Réfléchir à la façon d'améliorer votre implémentation existante.

Processus de TDD :

Il y a 5 étapes dans le flux TDD :

1. Lisez, comprenez et traitez la demande de fonctionnalité.
2. Traduisez l'exigence en écrivant un test unitaire. Si vous le lancez, le test unitaire s'exécutera et échouera car aucun code n'est encore implémenté.
3. Rédigez et implémentez le code qui répond à l'exigence. Exécutez tous les tests et ils devraient réussir, sinon répétez cette étape.
4. Nettoyez votre code en le refactorisant.
5. Rincez, faire mousser et répéter.



Les frameworks pour faire du TDD :

Il existe beaucoup de framework pour faire du TDD. Je vais vous en citer quelques une :

JUnit : un framework de test de pour le langage JAVA.

PHPUnit : un framework de test pour Php

Selenium : un framework de test pour Python

Testing : un framework de test pour JAVA

Karma : un framework de test pour Javascript

BDD (Behavior Driven Development) :

Le BDD est une approche de test dérivée de la méthodologie Test-Driven Development (TDD). Dans BDD, les tests décrivent le comportement du système. Dans la plupart des cas, l'approche *Given-When-Then* est utilisée pour écrire des cas de test.

Pourquoi faire du BDD :

Le BDD a de nombreux avantages :

- Encourager la collaboration entre les rôles pour construire une compréhension partagée du problème à résoudre.
- Augmenter la rétroaction et le flux de valeur.
- Produire une documentation système qui est automatiquement vérifiée par rapport au comportement du système.

Pour ce faire, il faut concentrer le travail collaboratif sur des exemples concrets et réels qui illustrent la façon dont nous voulons que le système se comporte. Nous utilisons ces exemples pour nous guider du concept à la mise en œuvre, dans un processus de collaboration continue.

BDD et agile :

Le BDD ne remplace pas votre processus agile existant, il l'améliore.

Le BDD est comme un ensemble de plug-ins pour le processus existant. Il rendra l'équipe plus à même de tenir les promesses de l'agilité : des versions rapides et fiables de logiciels fonctionnels qui répondent aux besoins évolutifs de l'organisation, nécessitant des efforts de maintenance et de la discipline.

BDD en trois (3) pratiques :

Essentiellement, le BDD est un processus itératif en trois étapes :

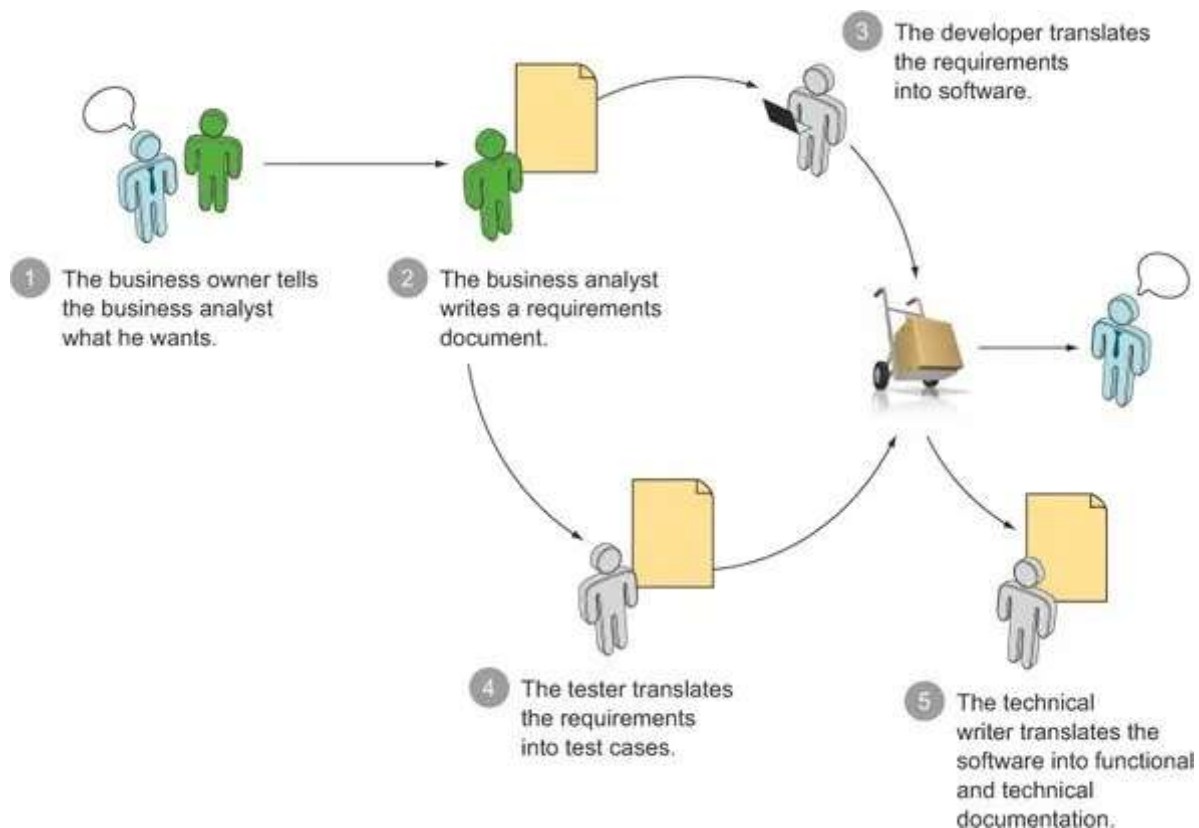
1. Tout d'abord, prenez un petit changement à venir dans le système une User Story et parlez d'exemples concrets de la nouvelle fonctionnalité pour explorer,

découvrir et convenir des détails de ce qui doit être fait.

2. Ensuite, documentez ces exemples d'une manière qui peut être automatisée et vérifiez leur accord.
3. Enfin, implémentez le comportement décrit par chaque exemple documenté, en commençant par un test automatisé pour guider le développement du code.

L'idée est de rendre chaque changement petit et d'itérer rapidement, en remontant d'un niveau chaque fois que vous avez besoin de plus d'informations. Chaque fois que vous automatisez et implémentez un nouvel exemple, vous ajoutez quelque chose de précieux à votre système et vous êtes prêt à répondre aux commentaires.

Nous appelons ces pratiques *Découverte*, *Formulation* et *Automatisation*.



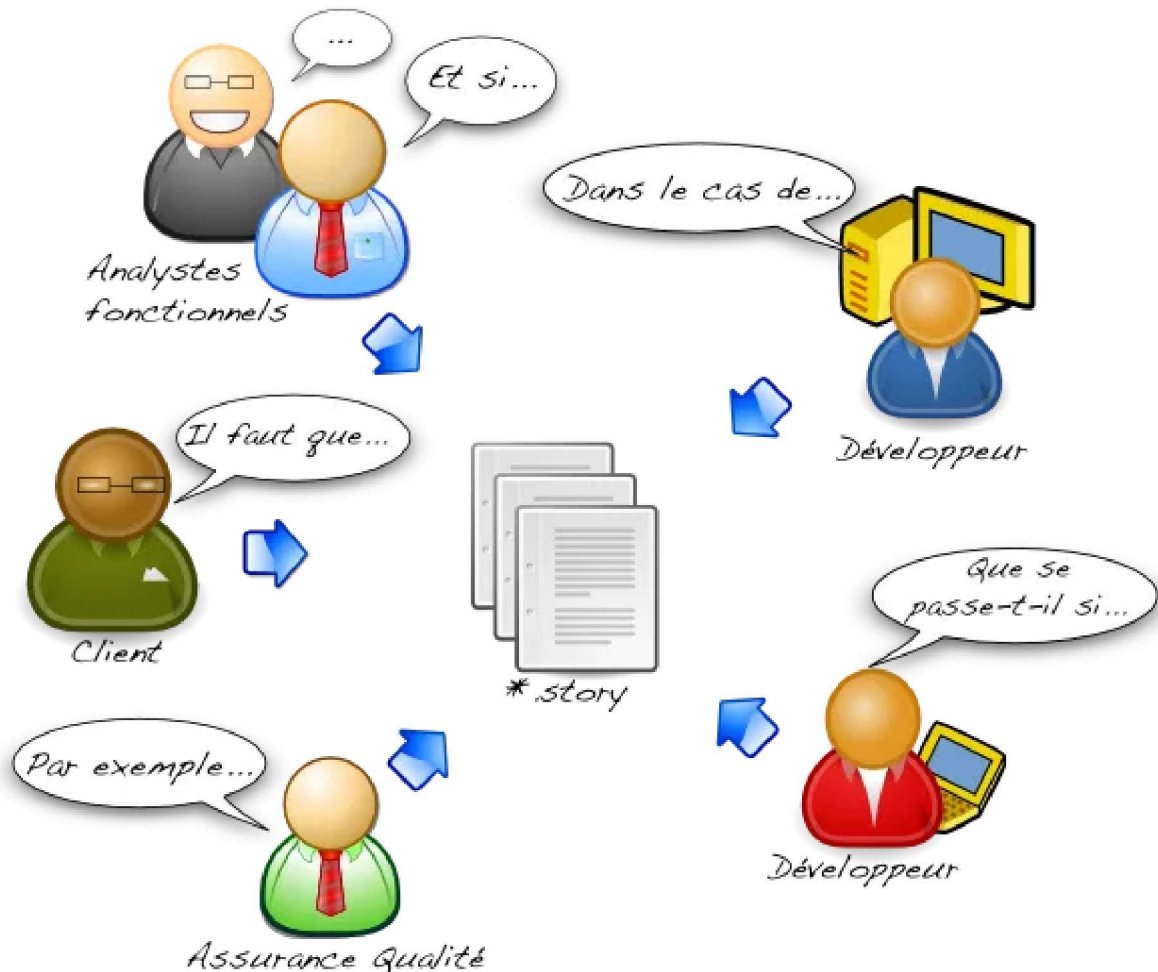
Définissez les choses à tester :

Exemple : Imaginons que l'on souhaite ajouter une fonctionnalité de commentaire pour l'application interne de Ouidou **StarOuids** pour permettre aux collaborateurs de Ouidou de laisser des avis.

Les fonctionnalités que nous pourrions tester :

- la taille limite.

- un commentaire vide.
- si tout le monde a le droit de commenter.
- etc.



La syntaxe BDD :

Les syntaxes seront souvent en anglais:

Étant donné : **Given**

Quand : **When**

Alors : **Then**

Et : **And**

Exemple :

Vos scénarios doivent décrire le comportement prévu du système, pas la mise en œuvre. En d'autres termes, il doit décrire le *quoi*, pas *comment*.

Par exemple, pour un scénario d'authentification, vous devez écrire :

```
Étant donné que je visite "/login"  
Quand j'entre "Abdou" dans le champ "nom d'utilisateur"  
Et j'entre "testeur" dans le champ "mot de passe"  
Et j'appuie sur le bouton "connexion"  
Alors, je devrais voir la page "Accueil"
```

Envisagez un style plus déclaratif :

Une façon de rendre les scénarios plus faciles à maintenir est d'utiliser un style déclaratif. Le style déclaratif décrit le comportement de l'application, plutôt que les détails de l'implémentation. Les scénarios déclaratifs se lisent mieux. Un style déclaratif vous aide à vous concentrer sur la valeur que le client veut obtenir, plutôt que sur les frappes qu'il utilisera.

Les tests impératifs communiquent des détails et, dans certains contextes, ce style de test est approprié. D'autre part, parce qu'ils sont si étroitement liés à la mécanique de l'interface utilisateur actuelle, leur maintenance nécessite souvent plus de travail. Chaque fois que la mise en œuvre change, les tests doivent également être mis à jour.

Voici un exemple de fonctionnalité dans un style impératif :

Fonctionnalité : Gestion des droits dans l'application starOuids

Scénario : Les collaborateurs n'ayant pas les droits

```
Étant donné qu'un utilisateur a un compte ouidou  
Et qu'il se connecte avec ses identifiants valides  
Et qu'il n'a pas un rôle admin  
Alors il doit juste pouvoir consulter.
```

Scénario : Les collaborateurs ayant les droits admins

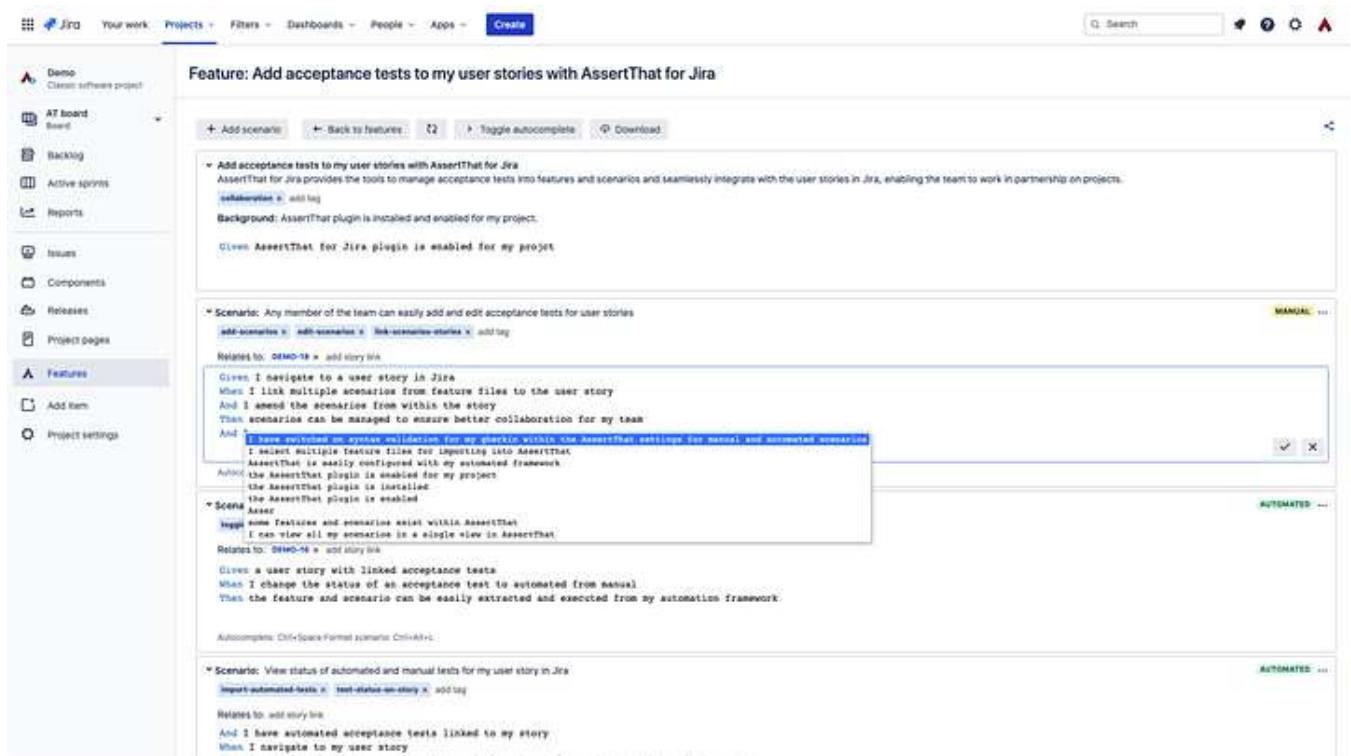
```
Étant donné qu'un utilisateur a un compte ouidou  
Et qu'il se connecte avec ses identifiants valides  
Et qu'il a un rôle admin  
Alors il doit pouvoir créer, consulter , modifier , supprimer
```

Faire du BDD et du TDD avec JIRA :

Il est possible de faire du BDD avec JIRA grâce au plugins Xray ou AssertThat.

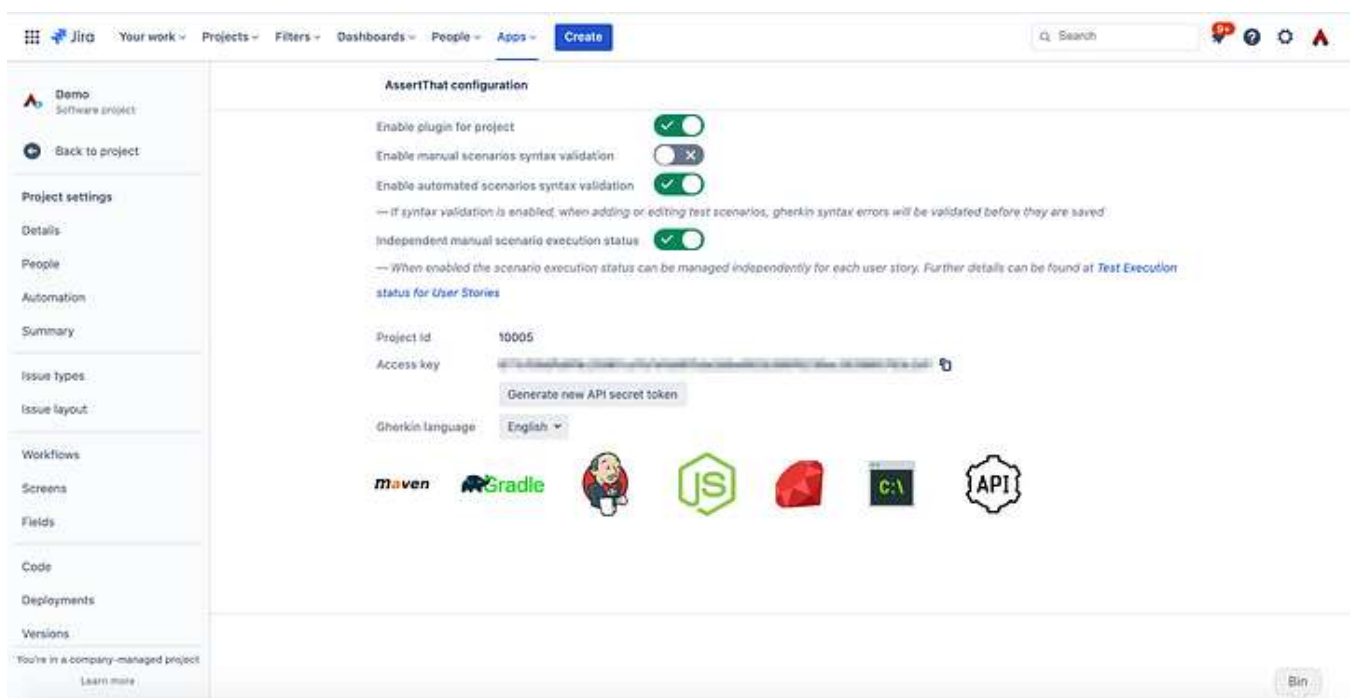
1. Écrire des scénarios BDD dans Jira pour une collaboration facile :

Avec AssertThat au cœur du processus BDD, vous pouvez écrire les comportements pour une compréhension commune des user stories.



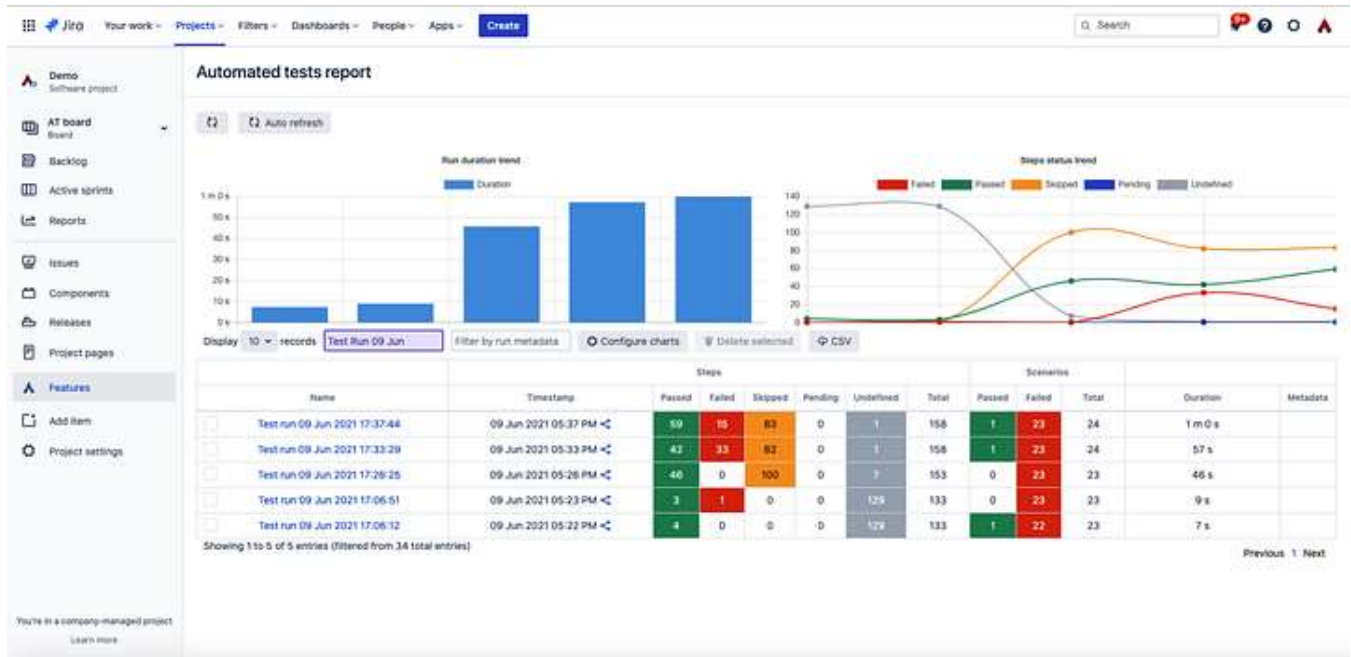
2. Intégration avec les frameworks d'automatisation des tests :

Accélérez l'automatisation des tests pour un retour rapide, avec les plugins, les API client et les intégrations (Jenkins, Gradle, Maven, Karate Framework et bien d'autres).



3. Suivre et analyser les résultats de l'automatisation des tests dans Jira

Pas besoin d'outils de reporting séparés, vos résultats de test sont réimportés dans Jira. Analysez facilement les tests ayant échoué, reliez les défauts et identifiez les tendances dans les résultats des tests automatisés pour une gestion des tests très efficace.



Différence entre TDD et BDD :

J'ai essayé de résumer dans un tableau 5 différences clé en TDD et BDD

Différences entre TDD et BDD		
	TDD	BDD
Concentrer sur	Livraison d'une fonctionnalité	Respecter le comportement attendu du système
Point de départ	Un cas de test	Une user story / scénario
Les participants	Équipe technique	Tous les membres de l'équipe, y compris le client
Difficulté de mise en œuvre	Relativement simple pour l'ascendant, plus difficile pour le descendant	La plus grande courbe d'apprentissage pour toutes les parties impliquées
Évite	Sur-ingénierie, faible couverture des tests et tests de faible valeur	Écart par rapport au comportement prévu du système

BDD vs TDD

Conclusion :

TDD et BDD sont deux méthodes qui peuvent parfaitement se combiner ensemble. Avec BDD, vous avez la possibilité de décrire le comportement de l'application et le TDD permettra de les transformer en des tests unitaires.

Pour aller plus loin :

L'importance d'écrire les tests en amont :

<p>Myth: "code must be tested by someone else"</p> <p>... and thus, by dedicated testers</p> <p>jp-lambert.me</p>	
--	--

Les différents outils de rédactions de spécifications exécutables :

Comparatif outils de rédaction de spécifications exécutables

Pour les démarches type ATDD, BDD, SpecByExample

jp-lambert.me

Vous vous demandez qui est Ouidou ? N'hésitez pas à nous contacter via contact@ouidou.fr ou visiter notre site <https://ouidou.fr>

Source :**TDD vs BDD vs ATDD : Key Differences | BrowserStack**

This guidepost aims to describe different testing methods or practices like Behavioral Driven Development (BDD)...

www.browserstack.com

BDD vs TDD - My Agile Partner Scrum

Quelle est la différence entre le TDD et la BDD (BDD vs TDD) ? Vous êtes nombreux à me poser cette question donc je...

blog.myagilepartner.fr

Bdd

Tdd

Agile

Development

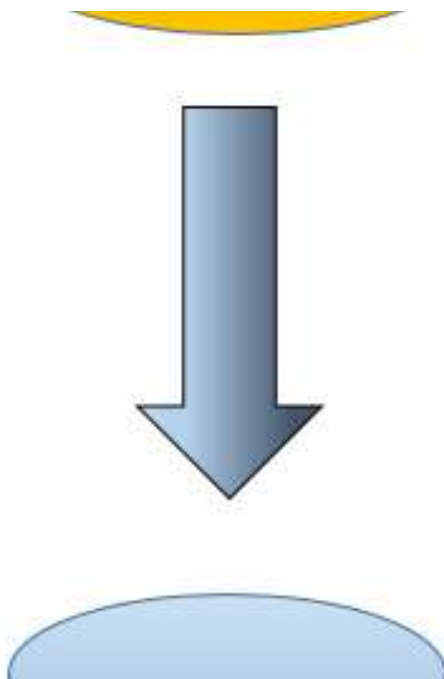
Test

[Follow](#)

Written by Abdourahmane Sow

5 Followers · Writer for Ouidou

More from Abdourahmane Sow and Ouidou



 Abdourahmane Sow in Ouidou

Comment migrer de SVN vers GIT à partir d'un dump en conservant l'historique.

Dans cette article nous allons voir comment faire une migration de SVN vers GIT.

3 min read · Jul 22, 2021



10





 Louis Jeanne-Julien in Ouidou

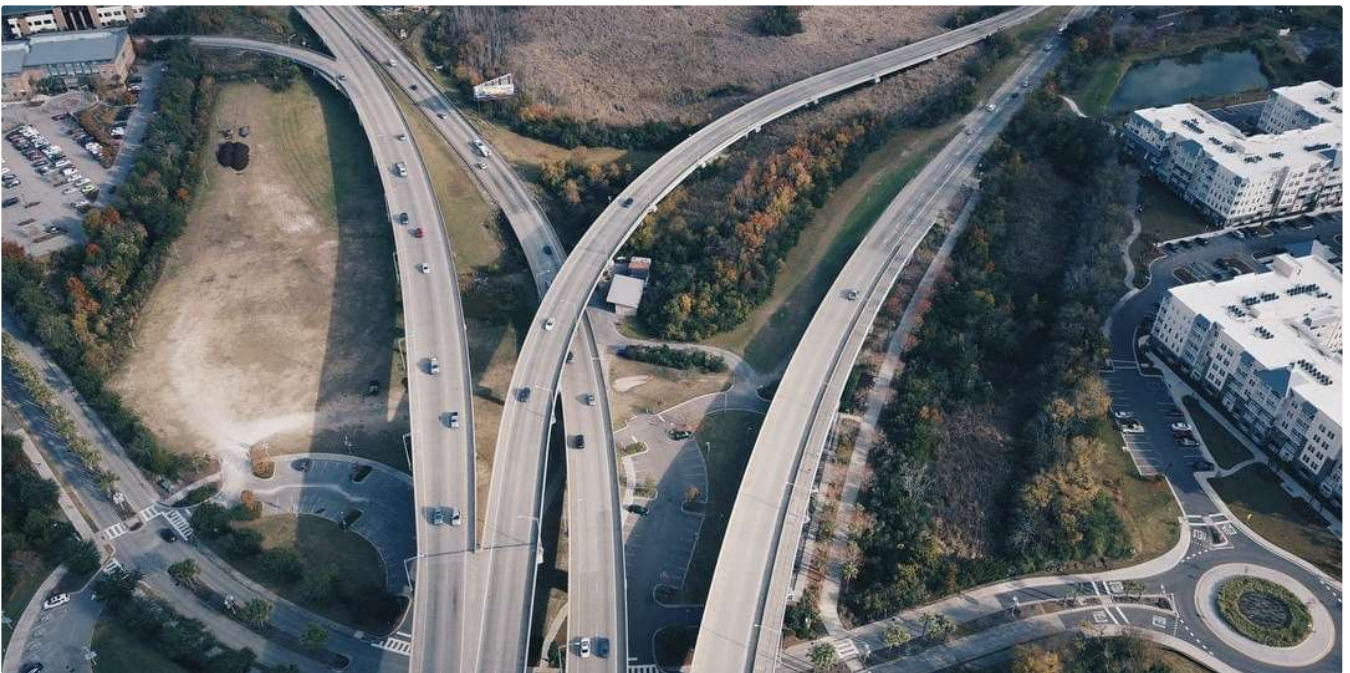
Spring Security (pour les nuls)


On est forcément passé un jour ou l'autre sur un projet impliquant Spring Security : le framework dont on dit beaucoup de bien, et qu'en...

14 min read · Mar 1, 2021



89



 Alexandre HAAG in Ouidou

À la découverte de Traefik

Dans cet article, je vous propose un tour d'horizon sur les fonctionnalités de Traefik v2.

6 min read · May 25, 2020



271



Lou Augey in Ouidou

Faire des tests unitaires dans les applications React avec Jest et Testing-Library

Dans cet article sera expliqué en quoi consiste les tests unitaires et comment les mettre en place dans une application React.js. Nous...

15 min read · Jan 26



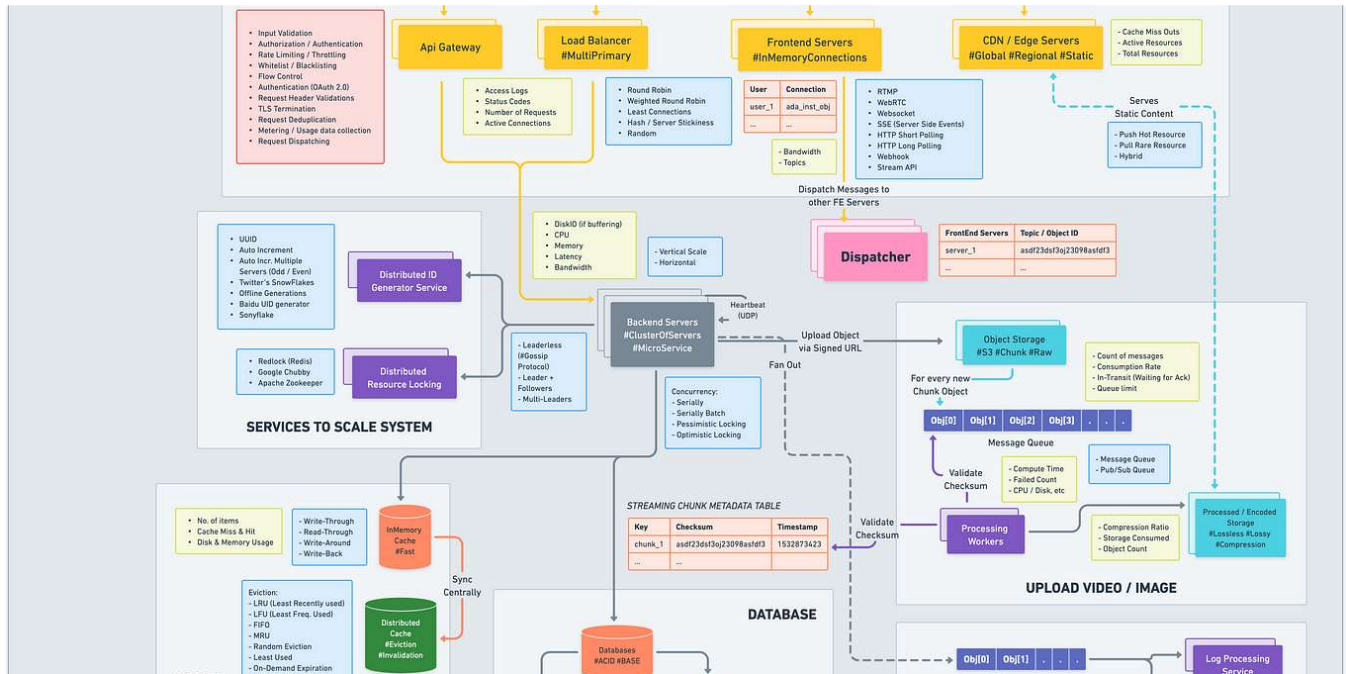
1



See all from Abdourahmane Sow

See all from Ouidou

Recommended from Medium



Love Sharma in Dev Genius

System Design Blueprint: The Ultimate Guide

Developing a robust, scalable, and efficient system can be daunting. However, understanding the key concepts and components can make the...

★ • 9 min read • Apr 20



4.1K



29




```
commit ffcf2c01b7ef612893529cef188cc1961ed64521 (HEAD -> master, origin/master, origin/bors/staging, origin/HEAD)
Merge: fc991bf81 5159211da
Author: iohk-bors[bot] <43231472+iohk-bors[bot]@users.noreply.github.com>
Date: Tue Nov 8 17:44:34 2022 +0000

Merge #4563

4563: New p2p topology file format r=coot a=coot

Fixes #4559.

Co-authored-by: Marcin Szamotulski <coot@coot.me>
Co-authored-by: olgahryniuk <67585499+olgahryniuk@users.noreply.github.com>

commit fc991bf814891a9349f22cf278632d39b04d4628
Merge: 5633d1c05 5cd94d372
Author: iohk-bors[bot] <43231472+iohk-bors[bot]@users.noreply.github.com>
Date: Tue Nov 8 13:07:58 2022 +0000

Merge #4613

4613: Update building-the-node-using-nix.md r=CarlosLopezDeLara a=CarlosLopezDeLara

Build the cardano-node executable. No default configuration.

Co-authored-by: CarlosLopezDeLara <carlos.lopezdelara@iohk.io>

commit 5159211da7a644686a973e4fb316b64ebb1aa34c
Author: olgahryniuk <67585499+olgahryniuk@users.noreply.github.com>
Date: Tue Nov 8 13:25:10 2022 +0200
```



Jacob Bennett in Level Up Coding

Use Git like a senior engineer

Git is a powerful tool that feels great to use when you know how to use it.

★ · 4 min read · Nov 15, 2022



6.4K



68



Lists



Good Product Thinking

11 stories · 143 saves



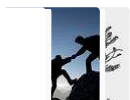
A Guide to OKRs – Objectives and Key Results

10 stories · 74 saves



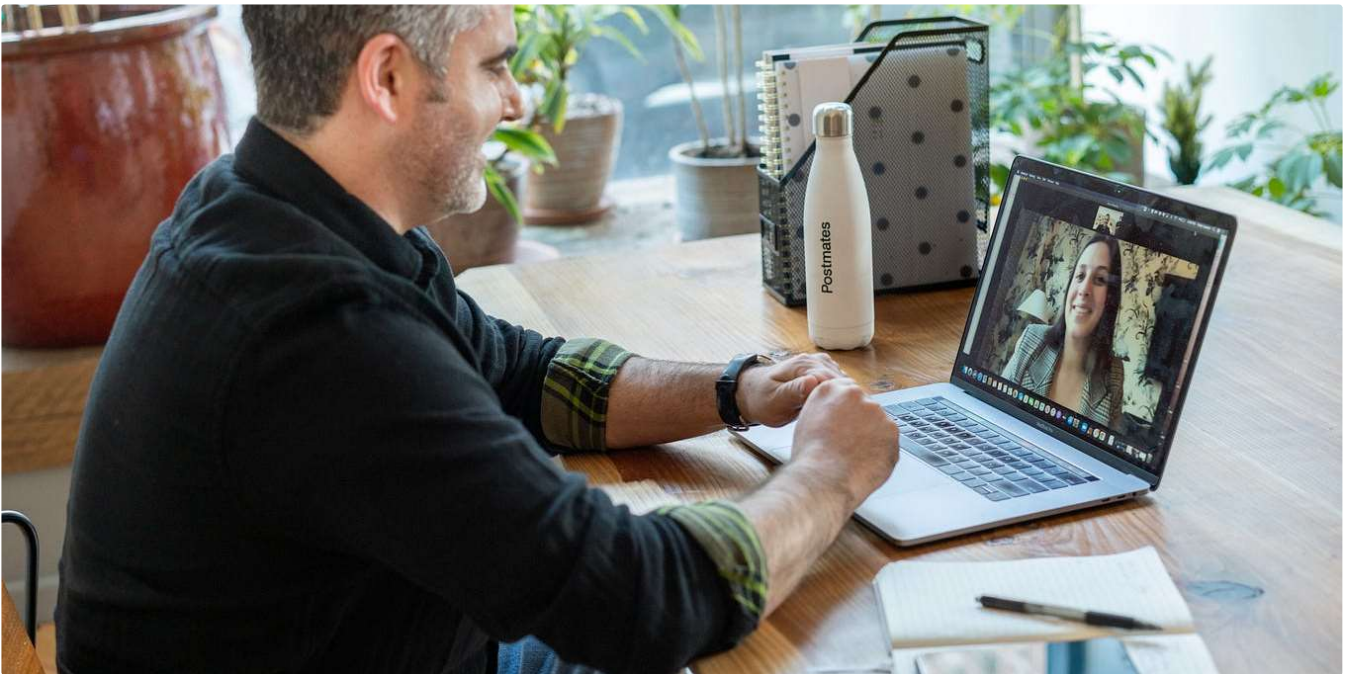
The New Chatbots: ChatGPT, Bard, and Beyond

13 stories · 24 saves



Leadership

31 stories · 66 saves



 Robbie Falck

QA Interviews and Tech Tests in 2023

Analysing the QA interview process from a candidate and interviewer perspective

★ • 9 min read • Feb 23



115



 Paul de Witt in Level Up Coding

What is the difference between Unit and Component Testing?

And why you should care.

★ · 5 min read · Jan 19

👏 58 💬



 Nick Wignall

4 Mental Habits that Cause Low Self-Esteem

#2: Worry about the future

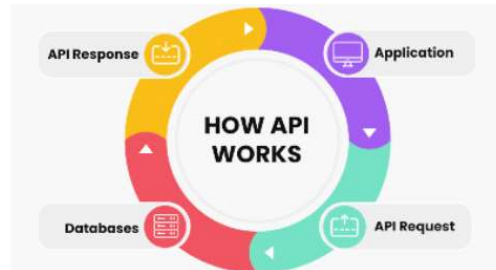
★ · 7 min read · 6 days ago

👏 1.7K 💬 47





API TESTING FOR MANUAL & AUTOMATION ENGINEERS



The Test Lead

QA API Testing Explained For Manual and Automation QA Testing

API testing in the QA field is a very desired skill set and knowing how to do so only makes you more marketable and valuable in the field...

★ • 3 min read • Jan 16



33



See more recommendations