

# Test Fonctionnels End To END (E2E)

Présenté par Mekouar Ayoub

# Introduction

Qui je suis

Qui vous etes

Ce dont j'ai besoin

Ce dont vous avez besoin

# Un bon cours

Moi	Vous
Investissement (prise de notes)	De la pratique
Réactivité	De bonnes explications (approfondies)
Questions même bêtes	Support de cours
	Interactivité

# Plan

## I- Les Tests

- Définition des tests end-to-end
- Autres types de test
- Les tests dans la planification agile
- Scénarios
- Conception de Scénarios
- Automatisation
- Environnement
- Remontée de bug
- TP

## II- La qualité ou QA

## III- TDD (Test Driven Développement)

# Définition des tests end-to-end

De bout en bout

Planification des tests : Spécifie les tâches principales(ticket), le planning (quand?, kanban) qui en découle ainsi que les ressources (assignments).

Conception des tests (documents, diagrammes) : Les spécifications de test(quand? , quoi ? Comment? Qui ? Critères de validation), la création de scénarios de test(étapes), l'analyse des risques et de leur utilisation. (régression, plantage, invalidation des critères)

Réalisation des tests : Exécuter les scénarios de test et rédiger les résultats des tests (document, statut completed)

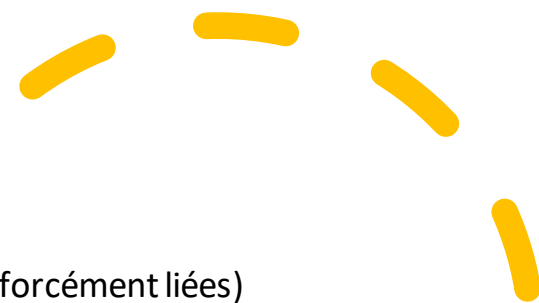
Analyse des résultats : Analyser les résultats des tests, les évaluer et effectuer des tests supplémentaires si nécessaire. (fait par PO, Responsable Qualité), Outils de Monitoring (faire du Coverage, Sonar (analyse du code, Lint), Performances (LightHouse, WebPack)

VERTICAL (en ordre) VS HORIZONTAL (EN vrac)

FONCTIONNALITES



# Comment définir une fonctionnalité

- 
- 1. Fonctionnalité
    - Qu'elle soit le plus large possible
    - Elle gère la même catégorie de données (forcément liées)
    - Intuition
  - 2. Pages
  - 3. Interactions
  - VENTES != CLIENTS
  - Connexion vs Inscription (Authentication)

## Autres types de test

Intégration : Tester en groupe les fonctionnalités

Unitaire : Tester chaque méthode (fonctions et procédures)

# Les tests dans la planification agile

Postes Agiles

Début ou fin ?

- Scénarios en sprint planning
- End to End pré validation => Pré-RELEASE
- Intégration : fin de sprint
- Tests Unitaires : Pendant les sprint

En tout cas les tests end to end c'est à la fin basés sur le plan de test (contient les 4 point de la diapo définition)  
Ou sous forme d'un bug bash

Valide le lancement définitif (E2E)

Qui fait les test ? => Préparé par le PO ou Responsable qualité  
=> Réalisé par les testeur QA, ou l'équipe de dev



# Scénarios de Test(un test d'intégration)

## EXHAUSTIF

Etape 1 : Identifier la fonctionnalité + Pages + Interactions

Etape 2 : Définir des conditions de validation par Interactions ou par Page.

Etape 3 : Ecrire les étapes

1. Se connecter
2. Aller à la page d'ajout d'un client
3. Remplir le formulaire
  - 3.1 le champ accepte minimum 3 car
  - 3.2 email respecte la regex
4. Cliquer sur submit

Etape 4 : Ecrire les résultats attendus => je valide les conditions de validation oui ou non. Si non => Remontée de bug

# Générés par CHAT GPT

- On va tous demander à chat gpt : comment écrire un scénario de test.



# Conception de Scénarios de Test

BUG BASH Ou un plan de test

Diagramme de séquence

Diagramme de flux d'activité

Issues

# Automatisation des test

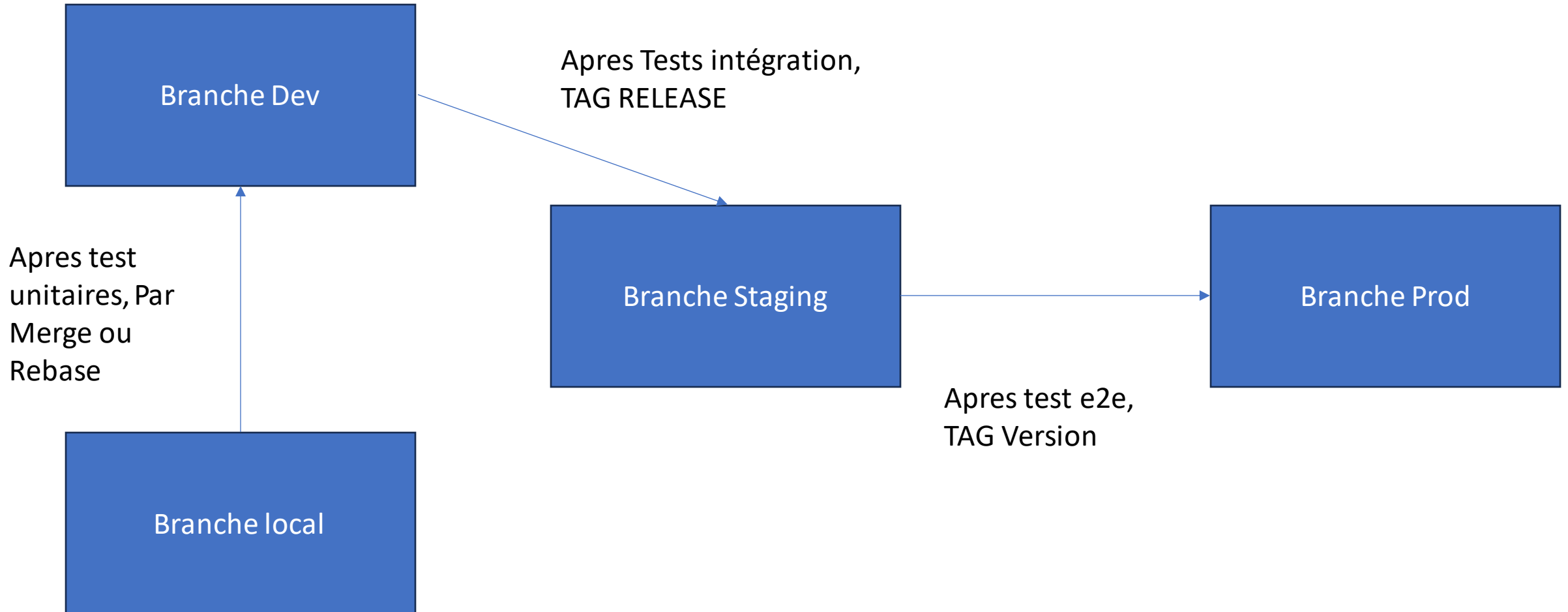
- Chaque langage a ses outils pour les test.
  - NodeJS =>
    - nightwatch pour les test e2e
    - Unitaire : JEST, Mockito
    - Integration : Mockito (genere des données de tests)
  - Java Spring Boot :
    - Spring boot test Selenium pour les test e2e
    - Unitaire : JUNIT, Mockito
    - Intégration : JUNIT, Mockito (genere des données de tests), cucumber.
  - C#
    - Selenium pour les test e2e
    - Unitaire : .net
    - Integration : .net
  - Kotlin
    - Kotlin Test pour les test e2e
    - Unitaire : JUNIT
    - Integration : JUNIT
  - Php Symphony
    - Selenium pour les test e2e
    - Unitaire : Jasmine, PHPUnit
    - Integration : Jasmine
  - CYPRESS

Environnement  
de test

La branche de  
staging

Quand est-elle  
changée ? RELEASE

# Les branches (Convention de Commit => Conventiionnal Commit)



# Remontée de bug

Statut => sur le kanban

Remonté par

Date de création

Type de bug : Régression, plantage, FRONT, BACK, BDD

Priorité : 1 à 3 vitesse de résolution

Sévérité : 1 à 3 a quel point ça impacte l'utilisation de l'application

Contexte technique (environnement, machine, navigateur ...)

Etapes de reproduction

Quel est le problème ? (avec screens)

A quoi vous vous attendiez ?

Critères de Validation





# TP

- Prendre une application déjà existante
- Faire un bug bash
- Faire au moins un diagramme de séquence
- Faire au moins un diagramme d'activité
- Ecrire un Scénario de test vertical
- Ecrire un Scénario de test horizontal
- Créer une branche de staging
- Exécutez le bug bash et faites une remontée de bug
- Bonus : intégrer un outil d'automatisation des tests (unitaire, intégration ou e2e)



Fonctionnalité	Oui	Non	Commentaire	Numéro de remontée de bug
I- Authentification				
a. Se connecter				
1. Champ email	X			
2. Champ mot de passe	X			
3. Lien vers les CGU et politique de confidentialité		X	Lien vers CGU ne fait rien	15
4. Click du bouton				
5. Lien mdp oublié				
6. Lien vers s'inscrire				
7. Apparition loader				
8. Menu cliquable				
9. Show Password				
10. Se souvenir de moi				
11. Bulle aide se souvenir de moi				
12. Bouton Annuler				
13. Redirection vers la home page				
b. Créer un compte				
1. Champ email, mdp,				

# Scénario de test horizontal étape 3

- Formulaire de contact

- |               |          |          |                                    |
|---------------|----------|----------|------------------------------------|
| ○ Nom Prénom  | Sujet    | Message  | Bouton envoyé                      |
| ○ (min 3 car) | (Requis) | (Requis) | (message de succès et redirection) |



## Règles de validation des champs

- Un champ email doit suivre le regex
- Un champ nom ou assimilé doit avoir minimum 3 caractères
- Un MDP a au moins 8 car dont 1 maj 1 min 1 chiffre et 1 special
- Date de naissance au format JJ/MM/AAAA
- ...

# Plan

- I- Les Tests
- II- La qualité ou QA
  - Définition et ISO
  - Charte Critères de qualité
  - TP
  - Selenium
  - Données de test
  - Scripts de Préparation et Nettoyage
  - Rapport de test
  - Tableau de bord
  - Mise en place de feedback
- III- TDD



# Définition

- Correspond aux exigences du client
- Validation de tout les test d'intégration et scénarios utilisateurs
- Validation des Détection des problèmes
- Respect des normes (ISO)

# Référentiel ISO/IEC 25010

- Modèle de Qualité du Produit :
  1. Fonctionnalité (Functionality) : Capacité du logiciel à fournir des fonctionnalités répondant aux besoins spécifiés.
  2. Performance d'Efficacité (Efficiency Performance) : Utilisation des ressources par le logiciel par rapport à la quantité de rendement produite.
  3. Fiabilité (Reliability) : Aptitude du logiciel à maintenir un niveau de performance sous des conditions spécifiées pendant une période donnée.
  4. Compatibilité (Compatibility) : Aptitude du logiciel à s'interfacer avec d'autres composants ou systèmes spécifiés sans perte de fonctionnalité ou d'intégrité.
  5. Facilité d'Utilisation (Usability) : Effort nécessaire pour que les utilisateurs apprennent à utiliser le logiciel, le préparent pour une utilisation spécifiée et accomplissent leurs tâches.
  6. Satisfaction (Satisfaction) : Mesure de la satisfaction des utilisateurs lors de l'utilisation du logiciel.
  7. Compliance (Conformity) : Mesure dans laquelle le logiciel respecte les normes, les conventions ou les réglementations applicables.



TP

Reprendre chaque point du référentiel Iso et proposer une  
stratégie de vérification



# Selenium

- Se base sur les scénario utilisateur (diagramme d'activité)
  - Tests de non-régression
  - Automatisation des tests côté client (si il y a par exemple un long formulaire à remplir)
  - Test efficace de l'application dans des contextes d'exécution différents (dans le cas d'une application web : Firefox, Chrome, Edge...)
- Selenium no code :  
<https://chromewebstore.google.com/detail/selenium-ide/mooikfahbdckldjjndioackbalphokd>
- Exemple formulaire + exemple select



# Données de test

- Mocking
- <https://generatedata.com/#generator>

# Tp

- Depuis votre scénario de test vertical faites un record sur selenium
  - Testez ce record en utilisant du assert text ou autre
  - Dupliquer plusieurs fois votre test et veuillez à mettre des données erronées et vérifier les messages d'erreur
- 
- NB : vous pouvez generer vos données de test

# III -TDD

- Definition
- Autres Types de tests
- Preparation et nettoyage des tests unitaires
- Rapports de test

# Definition

Formulation en 2008 des lois de TDD

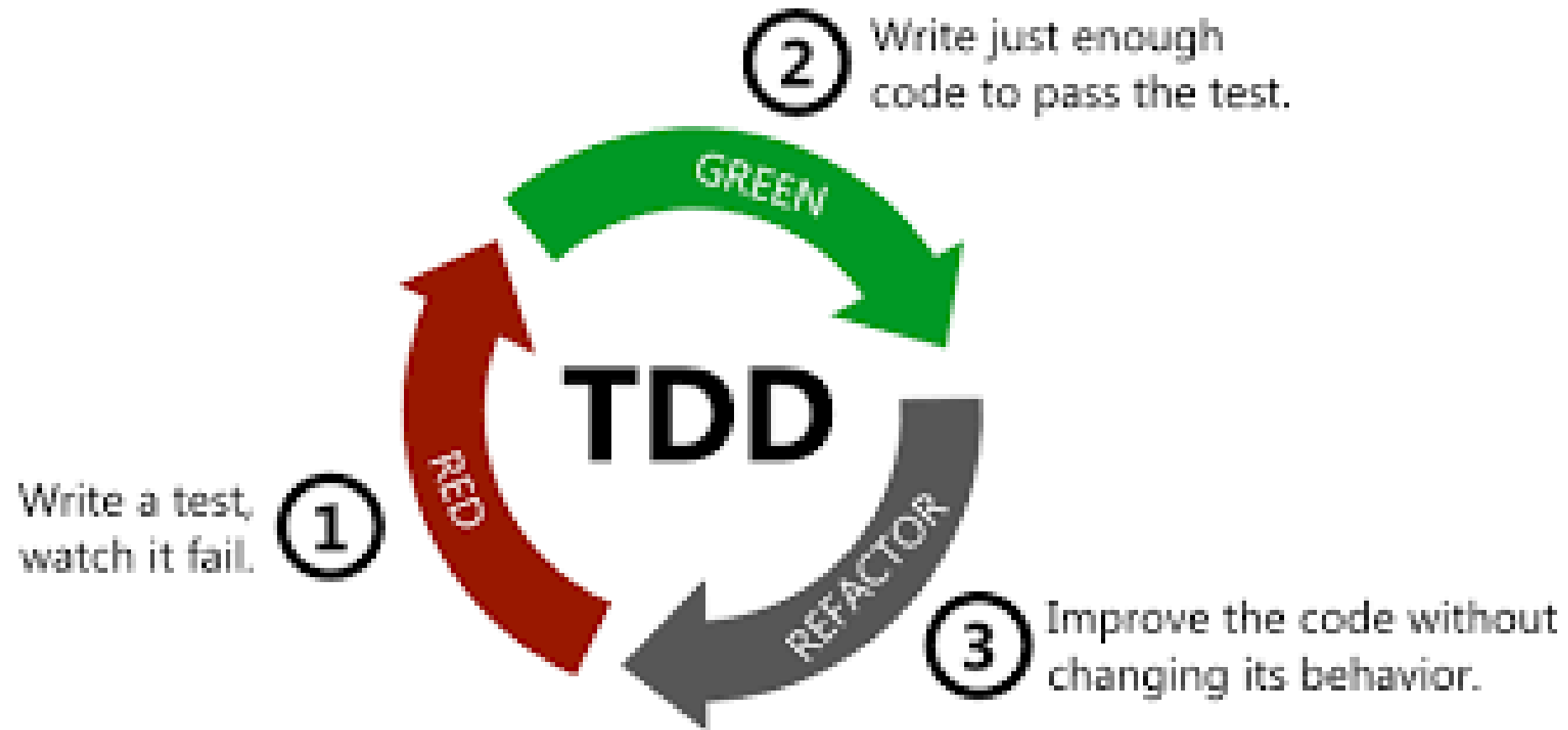
Dénomination ◆	Traduction française ◆	Version originale ◆	Note d'interprétation ◆
Loi n° 1	Vous ne pouvez pas écrire de code de production tant que vous n'avez pas écrit un test unitaire qui échoue.	"You may not write production code until you have written a failing unit test." <sup>2</sup>	C'est-à-dire qu'il n'est permis d'écrire du code de production que si un test unitaire est en échec <sup>3</sup> .
Loi n° 2	Vous ne pouvez pas écrire plus de code de test qu'il n'en faut pour qu'un test unitaire échoue, et ne pas compiler revient à échouer.	"You may not write more of a unit test than is sufficient to fail, and not compiling is failing." <sup>2</sup>	C'est-à-dire qu'il n'est permis d'écrire qu'un nouveau test unitaire en échec à la fois, et un test unitaire qui ne compile pas est déjà un test en échec <sup>3</sup> .
Loi n° 3	Vous ne pouvez pas écrire plus de code de production que nécessaire pour que le test unitaire actuellement en échec réussisse.	"You may not write more production code than is sufficient to pass the currently failing test." <sup>2</sup>	C'est-à-dire qu'il n'est permis d'écrire que du code de production permettant directement de faire passer le test unitaire précédent, ni plus ni moins <sup>3</sup> .

# Etapes du tdd

1. écrire un seul test qui décrit une partie du problème à résoudre ;
2. vérifier que le test échoue, autrement dit qu'il est valide, c'est-à-dire que le code se rapportant à ce test n'existe pas ;
3. écrire juste assez de code pour que le test réussisse ;
4. vérifier que le test passe, ainsi que les autres tests existants ;
5. remanier le code, c'est-à-dire l'améliorer sans en altérer le comportement, qu'il s'agisse du code de production ou du code de test.

# Avantages

- TDD permet d'éviter des modifications de code sans lien avec le but recherché, car on se focalise à chaque cycle sur la satisfaction d'un besoin précis, en conservant le cap du problème d'ensemble à résoudre.
- TDD permet d'éviter les accidents de parcours, où des tests échouent sans qu'on puisse identifier le changement responsable, ce qui aurait pour effet d'allonger la durée d'un cycle de développement.
- TDD permet de maîtriser le coût des évolutions logicielles au fil du temps, grâce à une conception du code perméable au changement.
- TDD permet de s'appropriier plus facilement n'importe quelle partie du code en vue de le faire évoluer, car chaque test ajouté dans la construction du logiciel explique et documente le comportement du logiciel en distillant l'intention des auteurs.
- TDD permet de livrer une nouvelle [version d'un logiciel](#) avec un haut niveau de confiance dans la qualité des livrables, confiance justifiée par la couverture et la pertinence des tests à sa construction.



# Tests de non regression

Correctif  
Complet  
Selectif  
Progressif  
Partiel  
unitaire


## COMMENT RÉALISER UN TEST DE NON RÉGRESSION ?

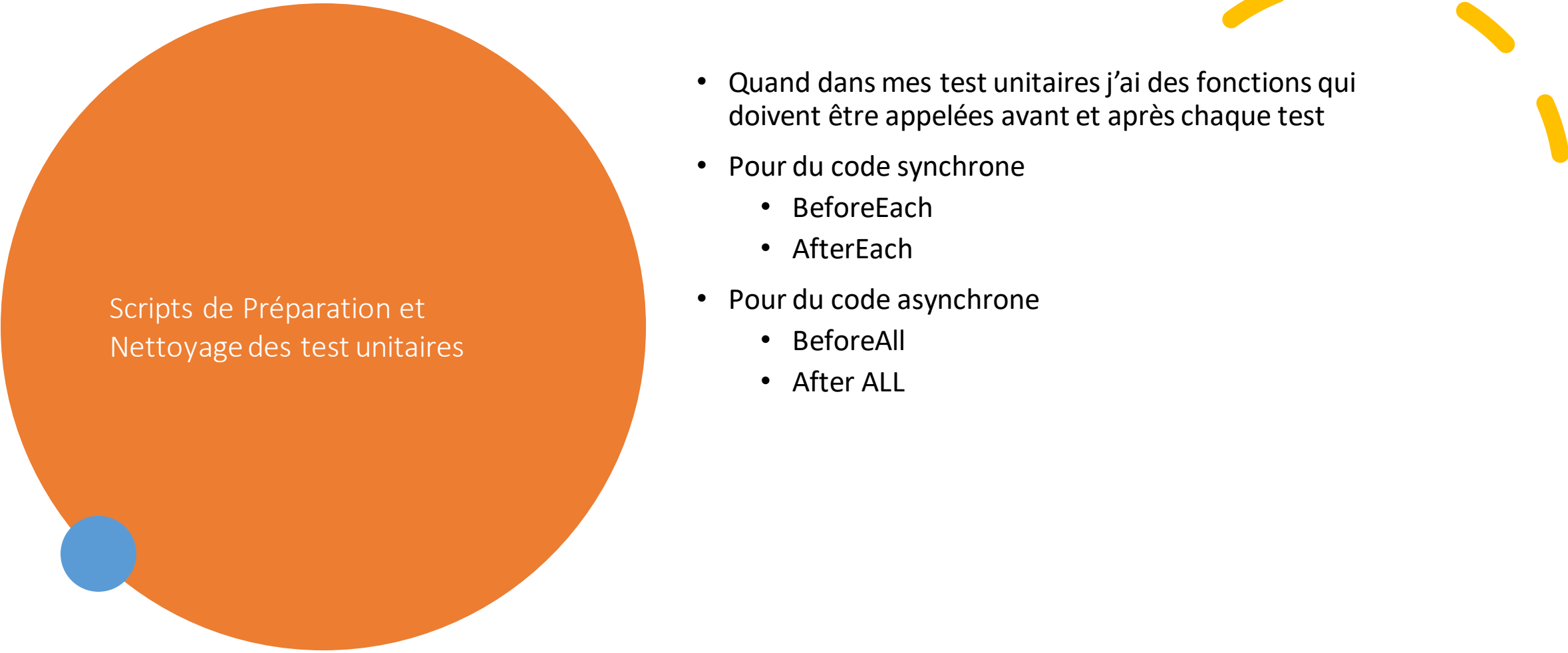






# Test de performance

- Test de charge
  - Test de stress
  - Spike test
  - Test d'endurance
  - Test de montée en charge
- 




## Scripts de Préparation et Nettoyage des test unitaires

- Quand dans mes test unitaires j'ai des fonctions qui doivent être appelées avant et après chaque test
- Pour du code synchrone
  - BeforeEach
  - AfterEach
- Pour du code asynchrone
  - BeforeAll
  - After ALL



# Rapport de test

- 
- Le nombre de cas de tests exécutés
  - Le nombre de cas de test réussis
  - Le nombre de cas de test échoue
  - Pourcentage de réussite
  - Pourcentage d'échec
  - Commentaires
  - Nombre total de bug
  - Statut des bugs (ouverts, fermés, en réponse)
  - Nombre de bugs ouverts, résolus, fermés
  - Répartition par sévérité et priorité

Qualité du  
rapport de  
test



# Tableau de bord

## Mobile - site #2

15/01/2015 | [Protocole de test](#)

### Déroulé de l'étude



### Score utilisateurs

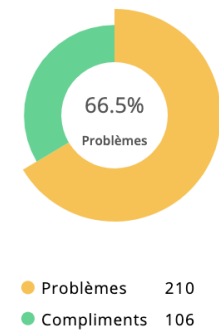
Moyenne des scores utilisateurs perfection : 3,6



[Commentaires des utilisateurs >](#)

### Proportion de problèmes

Moyenne des tests perfection : 51% de problèmes





# Mise en place de feedback

- Enquêtes sur les désabonnements
  - Score d'effort client (CES)
  - Score de [satisfaction du client](#) (CSAT)
  - Score d'adéquation produit-marché (PMF)
  - Net Promoter Score
  - Enquête de desabonnement
  - Enquêtes longues
- 
- Lien : <https://userpilot.com/blog/fr/best-customer-feedback-tools-saas/>

CES




CSAT

Rate your conversation

☹️ 😐 😐 😐 😍

Anita was excellent, thanks for the help

✈️

 We run on Intercom




PMF

**How would you feel if you  
could no longer use our  
product?**

- ☐ Very disappointed
- ☐ Somewhat disappointed
- ☒ Not disappointed

NPS

Ask me later

userpilot

How likely it is that you would recommend ABC Software to a friend?

0

1

2

3

4

5

6

7


8

9

10

Not likely

Extremely likely

 Userpilot

# Enquêtes longues

[Company Name]

[Street Address]

[City, ST ZIP Code]

## Customer Service Survey

How can we improve?

Please take a moment to help us improve your experience at [Company Name]. When you're done, please drop the questionnaire in the blue box at the front of the store.

### Product Quality

How often do you come to [Company Name]?

Every day  
4 or 5 times a week  
2 or fewer times a week  
First time

How would you rate our [type of products]?

Consistent, high quality  
Generally good  
Quality varies daily  
Poor quality

What do you typically purchase?

[product 1]  
[product 2]  
[product 3]  
[product 4]  
[product 5]  
Other

How would you rate our [type of products]?

Consistent, high quality  
Generally good  
Quality varies daily  
Poor quality

### Service and Environment

How long did you wait for your order to be taken?

Immediate service  
Less than 1 minute  
1 to 3 minutes  
More than 3 minutes

How long did you wait for your product after ordering?

Less than 1 minute  
1 to 3 minutes  
3 to 5 minutes  
More than 5 minutes

How would you rate the staff?

Friendly and helpful  
Average  
Varies on each visit  
Poor service

Was the store clean and inviting?

Yes  
No

### Additional Comments

---

---

---

---

### About You (optional)

Name 

---

Address 

---

Phone 

---

Email 

---

May we add you to our mailing list, which offers news and exciting promotions? ☐ Yes ☐ No

*Thank you for your participation!*

# TP

- Proposez un type de test de non regression et dire quand il sera effectué en écrivant un process respectant les 5 étapes
- En fonction des bug et tests des premiers tp proposez un rapport de test
- Ecrire un texte explicatif des types de feedback que vous aimeriez mettre en place et proposez des exemples
- Faire un bout de code en utilisant la méthodologie TDD : prendre des screens à chaque étape