INF5190 - Tests d'applications Web

Jean-Philippe Caissy

18 septembre 2019

Tests d'applications Web

Un code non testé est fondamentalement brisé

Les tests d'applications Web assurent une assurance qualité de l'application testé.

Il existe trois niveaux de tests :

- Tests unitaires
- Tests fonctionnelles
- Tests d'intégrations

Suite de tests

Le différents tests d'une application sont regroupés en une suite de tests.

Ils sont regroupés dans les trois niveaux (unitaire, fonctionnelles et intégration).

Une suite de test efficace possède les 4 caractéristiques suivantes :

1. Rapide
2. Complet
3. Fiable
4. Hermétique

Suite de tests

1. Rapide

- Plus un test est rapide, plus souvent il est exécuté lors du développement
- Les tests devraient être à chaque changement au code
- Une boucle de rétroaction rapide (*feedback loop*) aide au développement
- Si un test devient trop lent, il pourrait être ignoré

Suite de tests

2. Complet

- La suite de test devrait couvrir tout le code de l'application
- Supprimer du code de l'application doit faire échouer les tests
- Si la couverture n'est pas complète, il est difficile d'avoir confiance aux changements

Suite de tests

3. Fiable

- Les tests ne devraient pas aléatoirement échoués
- Des faux positifs minent a confiance à la suite de tests
- Les erreurs intermittents sont difficile à diagnostiquer

Suite de tests

4. Hérmétique

- Les tests roulent indépendant d'eux autres.
- Chacune des suites de test est responsable pour s'initialiser et libérer les ressources à la fin
- Les tests ne sont pas dépendant d'un ordre d'exécution

Types de tests

Tests unitaires

- Tests de premier niveau
- Sert à tester des unités spécifique de l'application web
- Ne doit tester et affirmer **qu'une seule** partie spécifique de l'application
- Ne nécessite pas l'instanciation complète de l'application Web
- L'interaction entre d'autres composantes est souvent remplacés par des *mocks*
- Les tests de modèle sont en grande partie des tests unitaires

Types de tests

Tests unitaires

Un test unitaire sert à tester :

- La fonctionnalité et la non-fonctionnalité d'une méthode
- La gestion des erreurs et des exceptions
- Les cas limites

Types de tests

Tests unitaires

Un test unitaire devrait être rapide, complet, fiable et hermétique :

- Pas de manipulation de fichiers ou de base de donnée
- Pas de communication réseau
- Pas d'effets de bords

Types de tests

Tests fonctionnels

- Tests de 2e niveau
- Niveau d'abstraction supérieur aux tests unitaires
- Vérifie une fonctionnalité du système
- Nécessite l'instanciation de l'application Web, car peut interagir avec plusieurs composantes
- Il s'agit souvent d'une seule requête à un contrôleur
- Les tests de contrôleurs et vues sont des tests fonctionnels

Types de tests

Tests fonctionnels

Un test fonctionnel sert à tester :

- La fonctionnalité d'une composante du système
- Un test fonctionnel va souvent couvrir l'équivalent de plusieurs tests unitaires

Types de tests

Tests fonctionnels

Un test fonctionnel peut :

- Manipuler des fichiers ou une base de donnée
- Faire des communications réseaux
- Avoir des effets de bords

Mais :

- Les effets de bords doivent être testés
- Le test fonctionnel doit nettoyer après celui-ci :
  - Toutes opérations de manipulation de fichiers doit être défaites
  - La base de donnée doit être remise dans l'état initiale

Types de tests

Tests d'intégrations

- Tests de 3e nievau
- Niveau d'abstraction similaire à une requête Web
- Vérifie que chacune des composantes s'intègre au système complet
- Un test d'intégration représente une ou plusieurs requêtes Web à l'application
- Exemple : Enregistrer un nouvel utilisateur sur l'application Web, et vérifier que celui-ci peut se connecter

Types de tests

Tests d'intégrations

Un test d'intégration va tester :

- L'envoie d'une requête HTTP à l'application Web
- Va couvrir du code touché par plusieurs composantes (modèle, vue et contrôleur)
- Peut faire plusieurs requêtes web dans un seul test

Types de tests

Tests de navigateurs

Un test de navigateur est un navigateur Web contrôlé par une API. C'est le test ultime d'une application Web.

- Utilisation d'outils qui automatise les actions d'un navigateur (aller à une adresse, cliquer sur un lien, etc)
  - C'est un vrai navigateur, seulement sasn "fenêtre"
- L'API fonctionne avec les éléments de la page. Par exemple :
  - Cliquer sur le lien dont le `id` est `admin`
  - Remplir les champs d'un formulaire et le soumettre
- On peut valider que les pages visités sont celles attendues, que les éléments Javascript de la page fonctionne, etc
- Les engins les plus utilisés sont ceux de Firefox ou Chrome

Types de tests

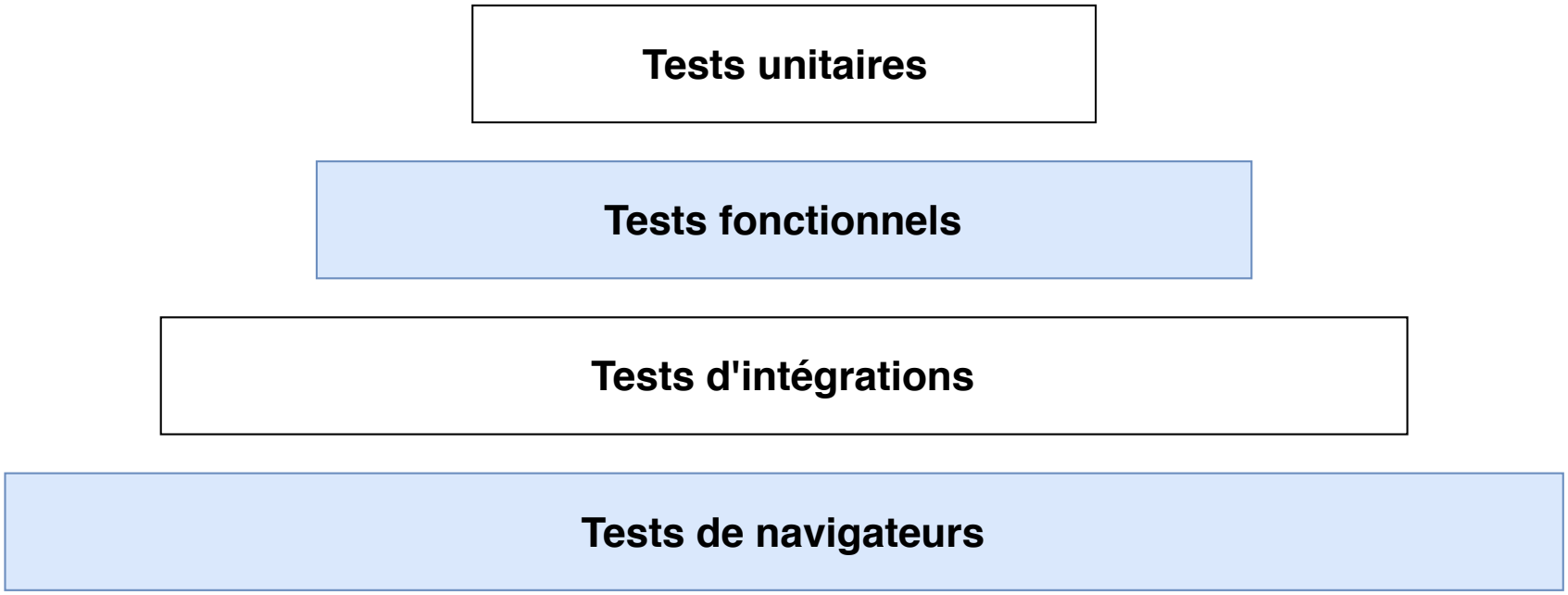


Figure 1: Pyramide des tests

Couverture de test

Afin de mesurer l'efficacité des tests, on peut analyser et mesurer le code qui n'a pas été testé.

```
$ pip install pytest-cov
[...]
$ python -m pytest --cov=poll
[...]

---------- coverage: platform linux, python 3.6.8-final-0
Name                    Stmts   Miss  Cover
-------------------------------------------
poll/__init__.py           50     15    70%
poll/models.py             90     14    84%
poll/services.py           10      3    70%
poll/views.py              22      4    82%
-------------------------------------------
TOTAL                     172     36    79%
```

Couverture de test

On peut avoir plus de détails sous forme d'une page HTML.

```
$ python -m pytest --cov-report=html --cov=poll
[...]
tests/test_integration.py .                                  [ 69%]
tests/test_unit.py ....                                      [100%]

----------- coverage: platform linux, python 3.6.8-final-0
Coverage HTML written to dir htmlcov


============= 12 passed, 1 skipped in 0.26s ==============
```

Les résultats sont disponible dans `htmlcov/index.html`

Couverture de test

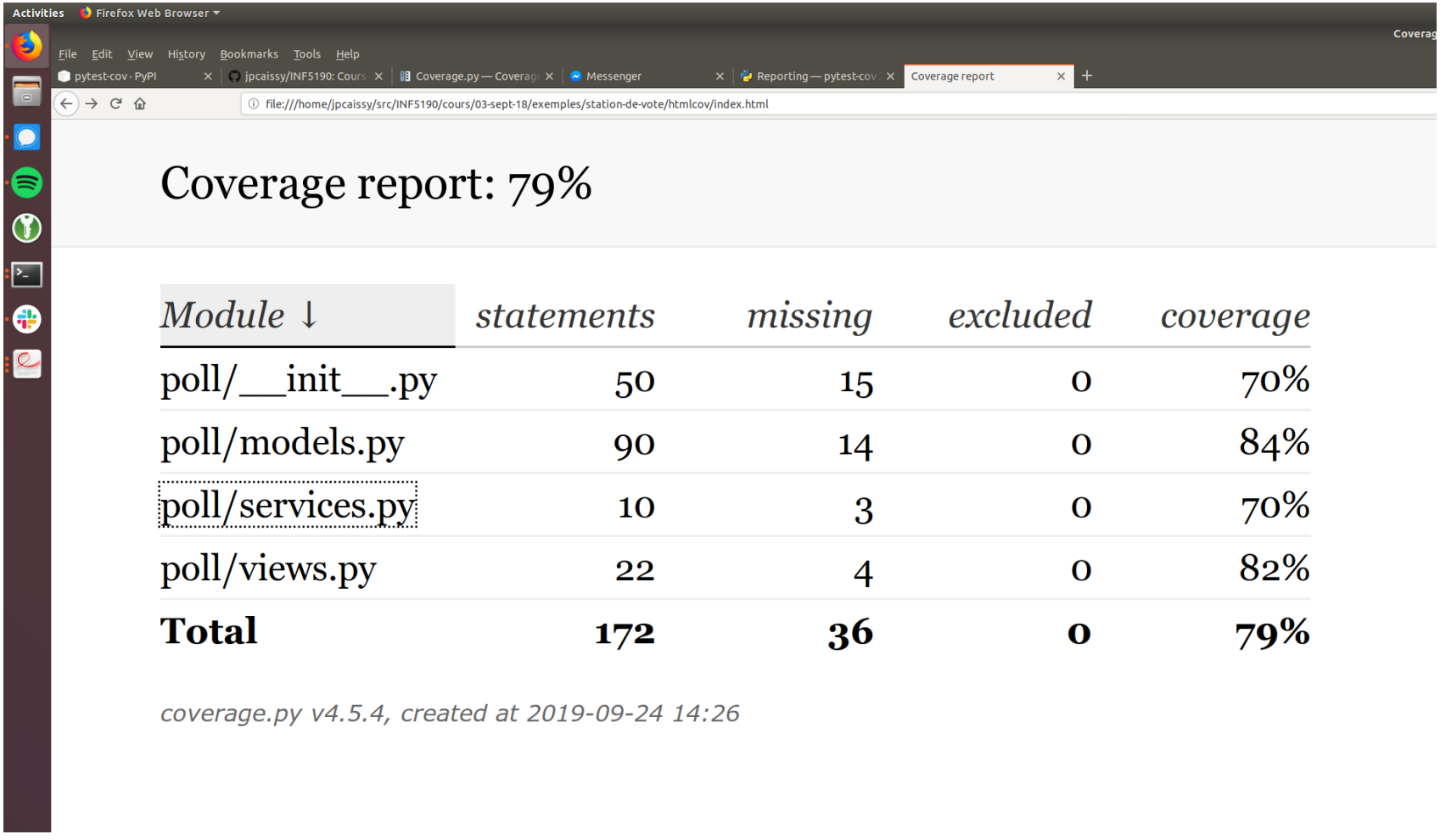


Figure 2: Rapport HTML de couverture des tests

Couverture de test

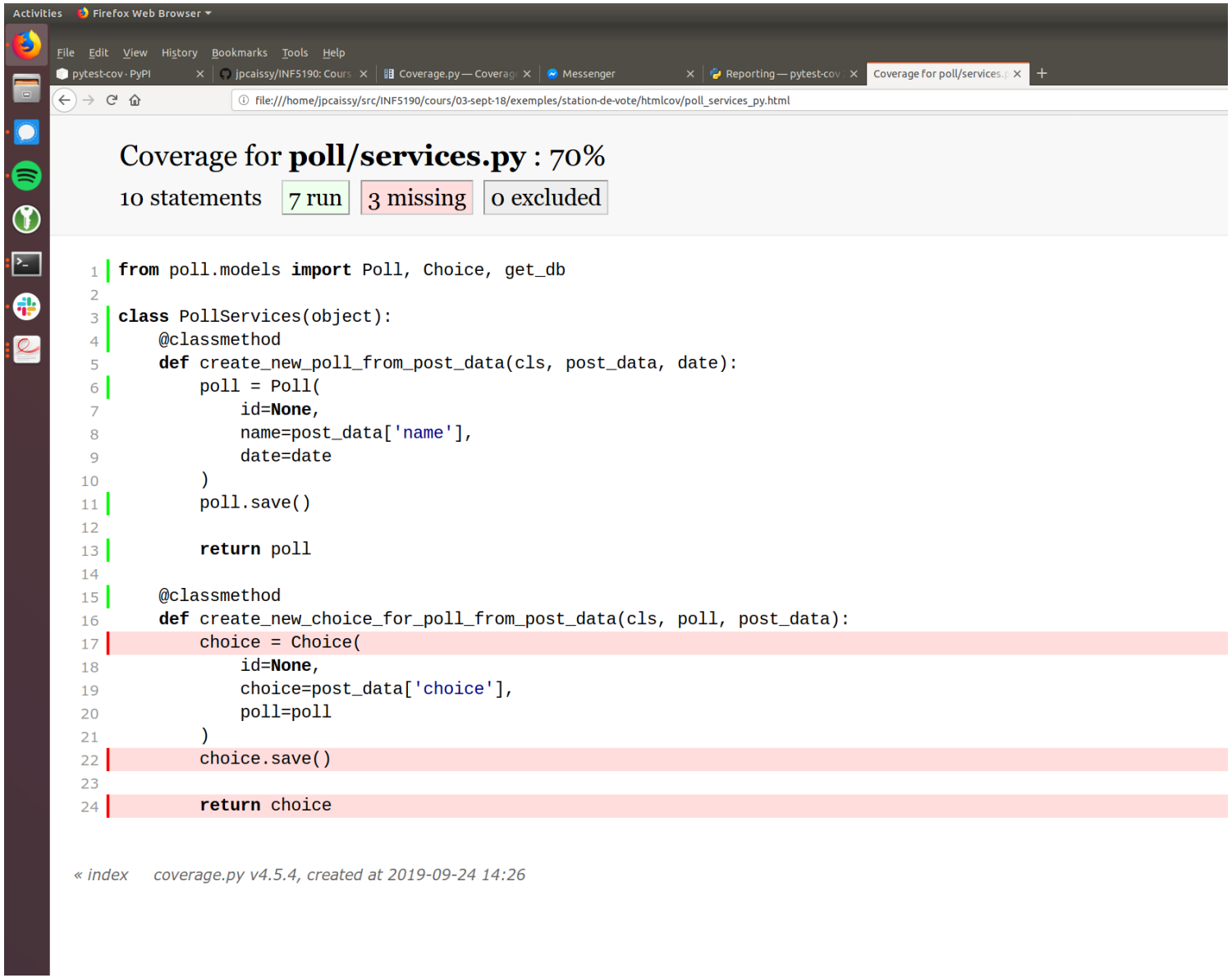


Figure 3: Rapport HTML de couverture des tests

Tests de performances

On utilise les tests de performance pour mesurer et valider la capacité d'une application Web

L'objectif est de répondre à la question suivante : combien d'utilisateurs est-ce que mon application peut servir concurrentiellement?

Tests de performances

Théorie des files d'attentes

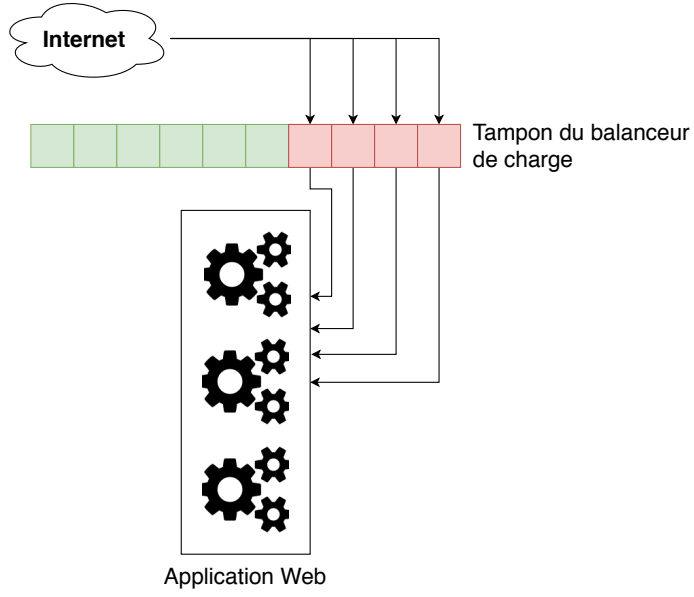
- Dans une application de type client-serveur, une file d'attente se crée à chaque fois qu'un client envoie une requête.
- À travers tout le système, plusieurs files d'attentes seront utilisés :
  - Le tampon (buffer) de la connexion TCP dans la carte réseau et l'OS
  - Le tampon du balanceur de charge
- Chaque composante d'une application Web a une capacité limite

Tests de performances

Théorie des files d'attentes

Requête dans une file d'attente (request queuing)

- Chaque requête Web va passer un certain temps (millisecondes) dans une file d'attente avant d'être traitée par l'application web
- Ce temps d'attente doit toujours être le plus bas possible

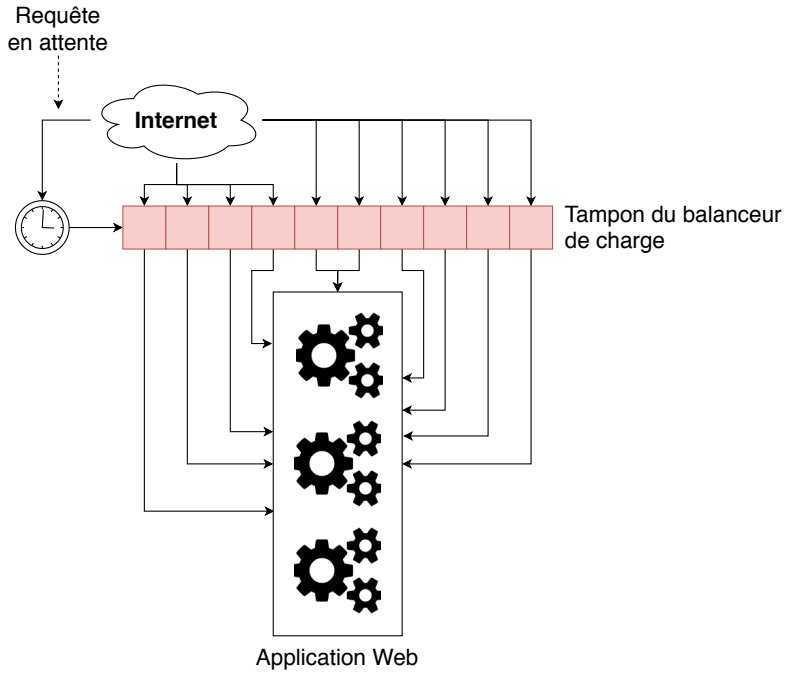


Tests de performances

Théorie des files d'attentes

Requête dans une file d'attente (request queuing)

- Lorsqu'une des files d'attentes du système est plein, il y a une attente de la requête (request queuing)



Tests de performances

Théorie des files d'attentes

Définitions

- Temps de réponse : Le temps perçu par le client (navigateur)
- Temps de réponse applicatif : Le temps que prends l'application Web à compléter une réponse
- Nombre de requête par unité de temps : Le nombre de requête concurrentes dans une intervale de temps
  - RPS : Requêtes par secondes
  - RPM : Requêtes par minute (600 RPM = 100 RPS)
- Temps d'attente d'une requête : La différence entre le temps de réponse et le temps de réponse applicatif
- Requêtes en attente : Lorsque le nombre de requête concurrente est supérieur à la capacité
- Capacité : Le nombre de requête concurrente qu'une application Web peut prendre sans avoir de requêtes en attente

Tests de performances

Théorie des files d'attentes

Loi de Little

Capacité d'un système est égaleà la fréquence moyenne d'arrivée X temps moyen passé dans le système

$$C = \lambda\tau$$

la capacité totale du système :

$$C$$

le nombre de requête par unité de temps :

$$\lambda$$

Le temps moyen passé dans le système :

$$\tau$$

Tests de performances

- Les tests de performances sont des outils qui simulent plusieurs personnes naviguant sur une application Web en même temps.
- Un test de performance peut durer quelques minutes à quelques heures
- Les informations suivantes sont capturés tout au long du test:
  - Nombre de requête concurrente
  - Temps de réponse moyen
  - Taux de succès et taux d'erreurs

Liens utiles

- Testing Rails
- How to apply Little's Law to validate performance test results
- Modern Web Automation With Python and Selenium