

Tests de performance

Avec Gatling

→ 24/03/2025

Agenda

Module	Date	Horaires	Salle
Module 1 Introduction à la méthodologie de test	04-mars	8h15-11h30	EF.202, Eiffel
Module 2 TDD / BDD	10-mars	8h15-11h30	SD.103, Bouygues
Module 3 Test Web Services : Postman	17-mars	8h15-11h30	Amphi e.068, Bouygues
Module 4 Automatisation Web Playwright	18-mars	13h45-17h	EF.110, Eiffel
Module 5 Automatisation App Mobile	21-mars	8h15-11h30	EF.202, Eiffel
Module 6 Test de performance : Gatling	24-mars	8h15-11h30	Amphi sc.071, Bouygues
Module 7 Accessibilité RGAA, Green, Data/IA	07-avr	8h15-11h30	TBD
Module 8 Evaluation	08-avr	13h45-17h	TBD

Logistique



Assiduité et émargement



Pauses



Interactivité et questions



Utilisation de laptops et smartphones



Evacuation

Sommaire

- 0. Introduction aux tests de performance
- 1. Initialisation projet Gatling (IntelliJ)
- 2. Développement Virtual User
- 3. Utilisation des données
- 4. Modèle de montée en charge
- 5. Cas pratique

0. Introduction aux tests de performance

Introduction

La gestion des risques au centre des tests de performance

Auditer la performance applicative

Capacité à identifier les points faibles et à obtenir un état des lieux du niveau actuel des performances

Identifier les Risques & les Exigences

Analyse technique et fonctionnelle des risques et des exigences de performance.

Formaliser les objectifs

Décrire les objectifs couvrants les Risques et les Exigences de performance.

Assurer la couverture

Associer les tests de performance aux objectifs pour assurer la couverture des risques et des exigences

Réaliser les tests de performance

Préparer et exécuter les tests. Analyser les résultats. Proposer des optimisations de la plateforme.

Amélioration continue

Analyser les incidents de performance et de disponibilité en production. Ajuster le modèle de conception des tests de performance

Quelques tristes exemples

Les objectifs de performance doivent être exprimés au **pic de fréquentation**/utilisation de l'application à tester.

Généralement les tests de performance permettent de s'assurer qu'en cas de pic, le service n'est pas dégradé.

09/08/2023

**DÉCLARATION DE BIENS IMMOBILIERS:
INDISPONIBLE EN PARTIE CET APRÈS-MIDI, LE
SITE DES IMPÔTS REFOCTIONNE**

14 mars 2023

**Passe Navigo : la plateforme
d'indemnisation de nouveau disponible
après une perturbation**

4 octobre 2023

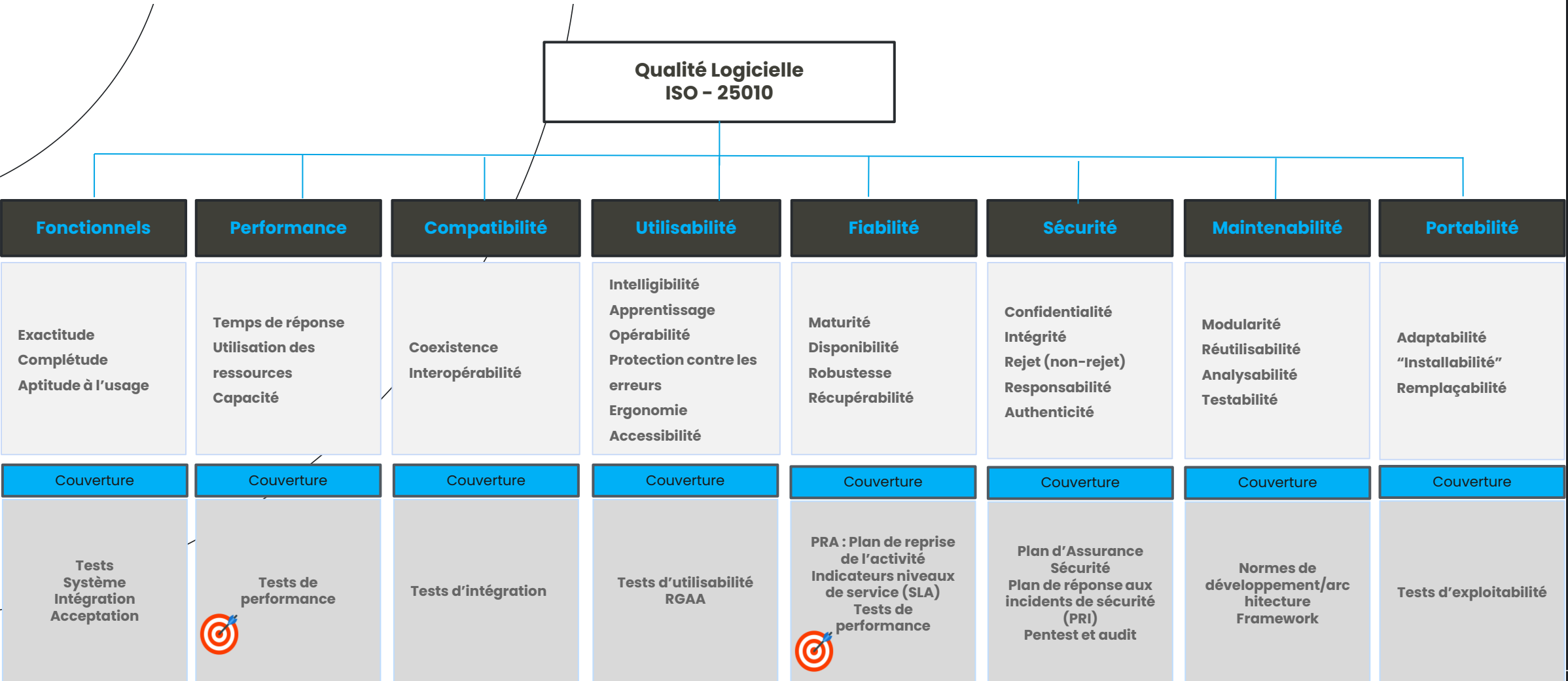
**Billets de train pour Noël : le site et l'application SNCF
Connect saturés ce matin, retour à la normale**

19/06/2023

**LE SITE DE MÉTÉO-FRANCE TOMBE EN PANNE
EN PLEINE SÉQUENCE ORAGEUSE**

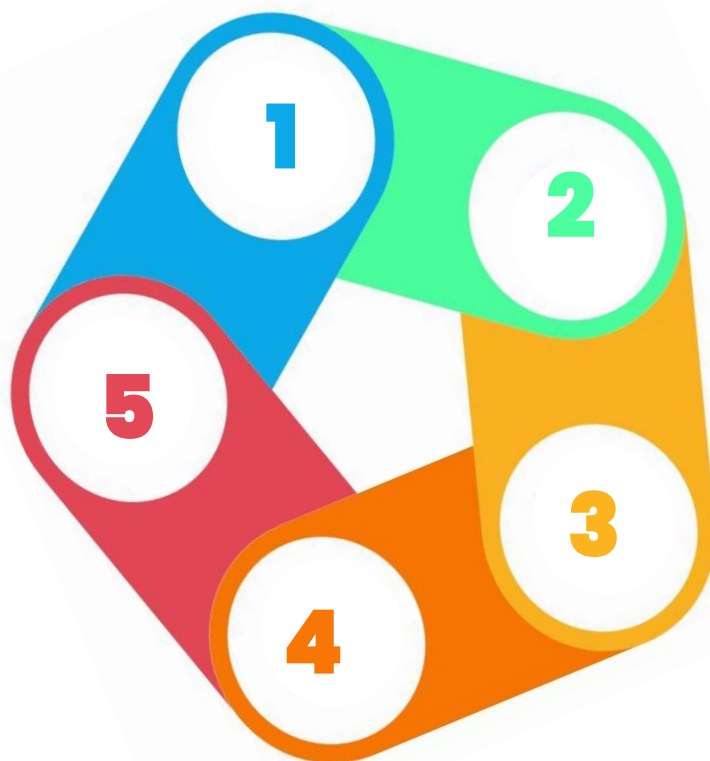


Vue d'ensemble : critères qualité logiciel ISO 25010



Implémenter les tests de performance au sein de l'organisation

vs)



1. Stratégie de test de performance

- Basée sur la gestion des risques
- Orientée exigences métiers
- Définie l'organisation, les rôles et les responsabilités

2. Performance By Design

- Audit du code
- Revue par les paires
- Bonnes pratiques pour un code performant
- Contrôle automatique du code intégré à la gestion de version

3. Gérer un projet de performance

- Test de performance Agile dans le sprint
- Validation produit
- Sécurisation de la transition en production

4. Identifier les incidents de performance en production

- Définir les métriques de performance
- Monitorer les systèmes et les processus métier
- Analyse de logs
- Alertes et reporting

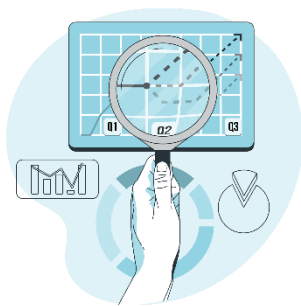
5. Mesurer et contrôler la performance et la disponibilité des applications

- Surveillance des parcours des utilisateurs réels
- Mesure de la disponibilité des processus métier
- Surveillance des applications et corrélation des événements trans-systèmes.
- Alertes et reporting

Processus de réalisation d'un test de performance



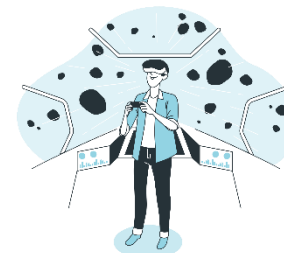
Parcours utilisateur



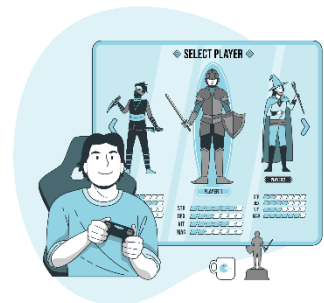
Gatling



Virtual User Script



Controller
Gatling



Load injector



Application Under Test



Monitoring

Conduire un projet de test de performance

1 PREPARER

- Analyse des risques
- Sélection des cas d'utilisation
- Définitions des objectifs de performance et des métriques associées
- **Identification des données**

2 RÉALISER

- Développement des scripts
- Implémentation des données dans les scripts et corrélations
- Préparation des données avec la volumétrie cible
- Définitions des modèles de montée en charge
- Mise en place de la plateforme d'injection de la charge
- Exécution des tests de validation globale (10% de la charge cible)

3 EXÉCUTER

- Alimentation des données
- Mise en œuvre des indicateurs de supervision
- Exécution des tirs de performance
- Analyse des résultats après chaque tir, débriefing et tuning

4 BILAN

- Analyse globale des exécutions et des bilans intermédiaires
- Rapport final avec conclusions et recommandation

Transfert de compétences

Types de test

Test de charge :

- Charge de travail supérieure à sa capacité normale.
- Simuler une situation de forte demande
- Capacité du système à gérer les pics et à maintenir un niveau de performance acceptable.

Test de stress :

- Conditions extrêmes, conditions aux limites
- Surcharge ou une saturation des ressources
- Capacité à résister à ces conditions sans tomber en panne ou réduire considérablement sa performance.

Test de montée en charge :

- Mesure la performance d'un système en augmentant progressivement la charge jusqu'à atteindre sa capacité maximale.

Test de endurance :

- Performance d'un système sur une période prolongée, généralement plusieurs heures ou jours
- Capacité à maintenir un niveau de performance acceptable sur une période de temps prolongée

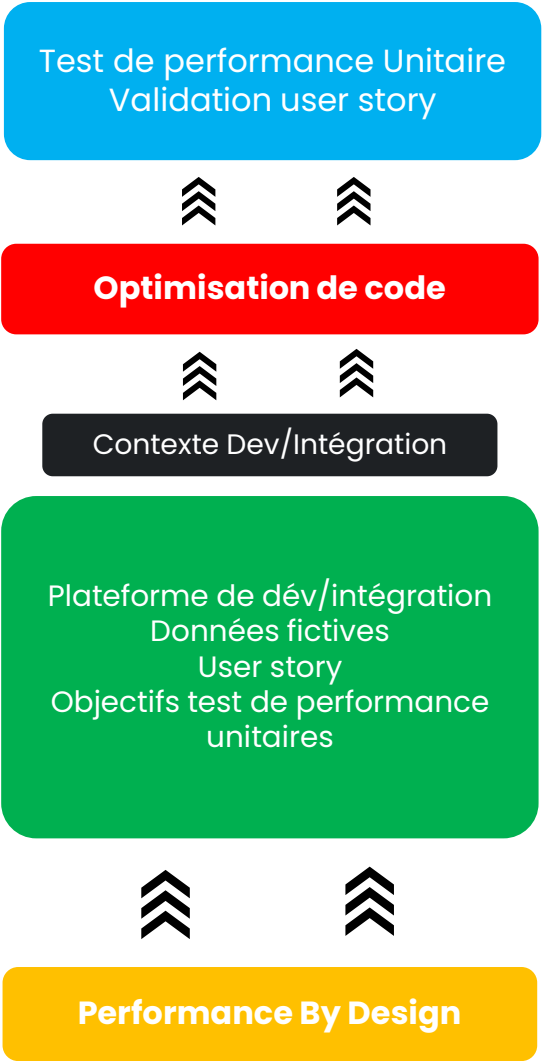
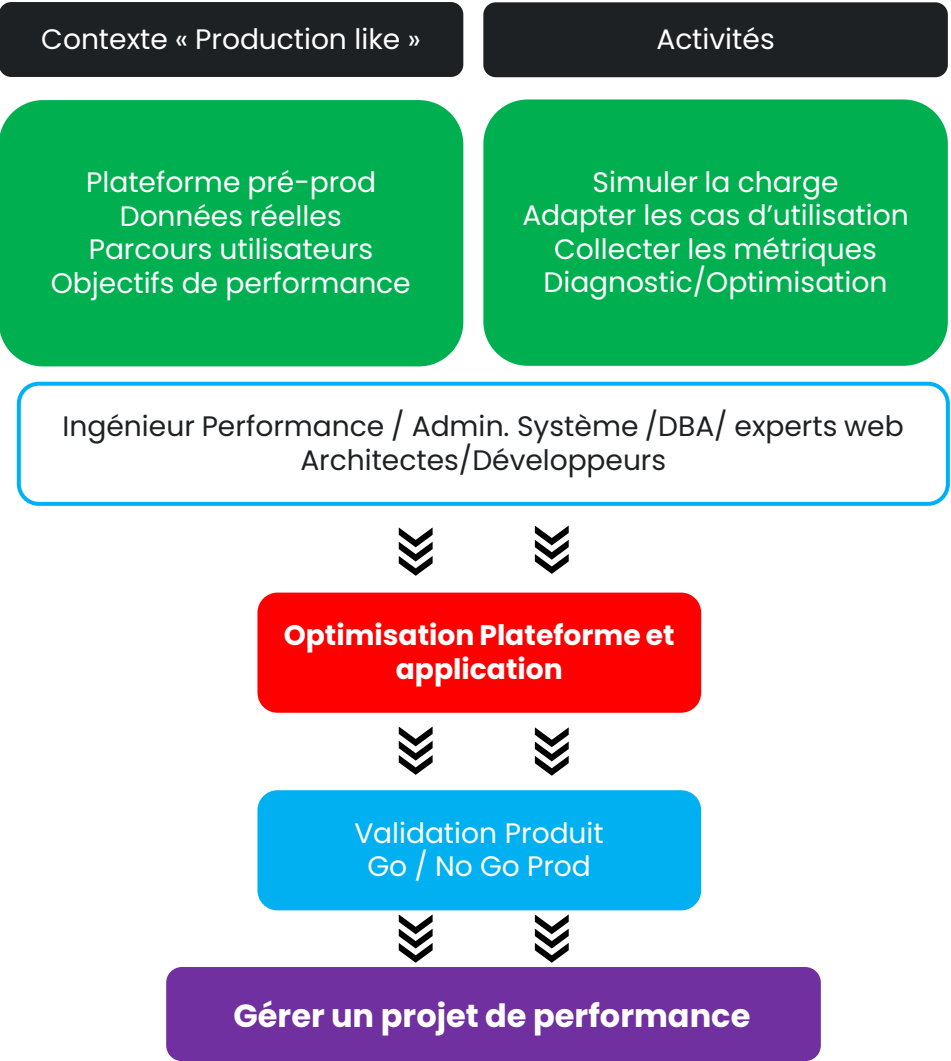
Test de configuration :

- Performance d'un système en modifiant sa configuration, telles que la mémoire, le processeur, le réseau, ou la base de données
- Impact de ces modifications sur la performance du système.

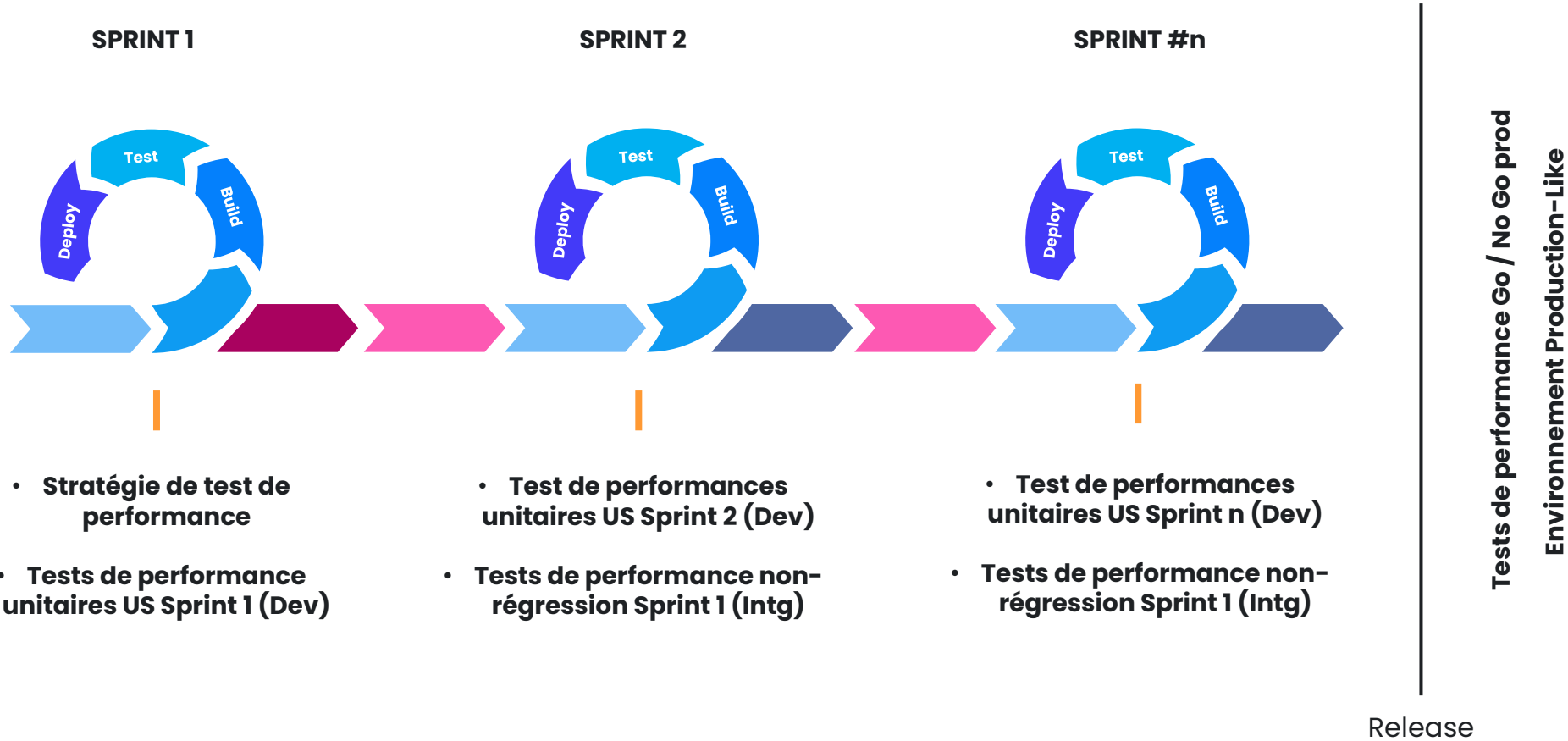
Test de volume :

- Performance d'un système avec une volumétrie de données importante.
- Capacité à maintenir un niveau de performance en cas d'augmentation du volume de données

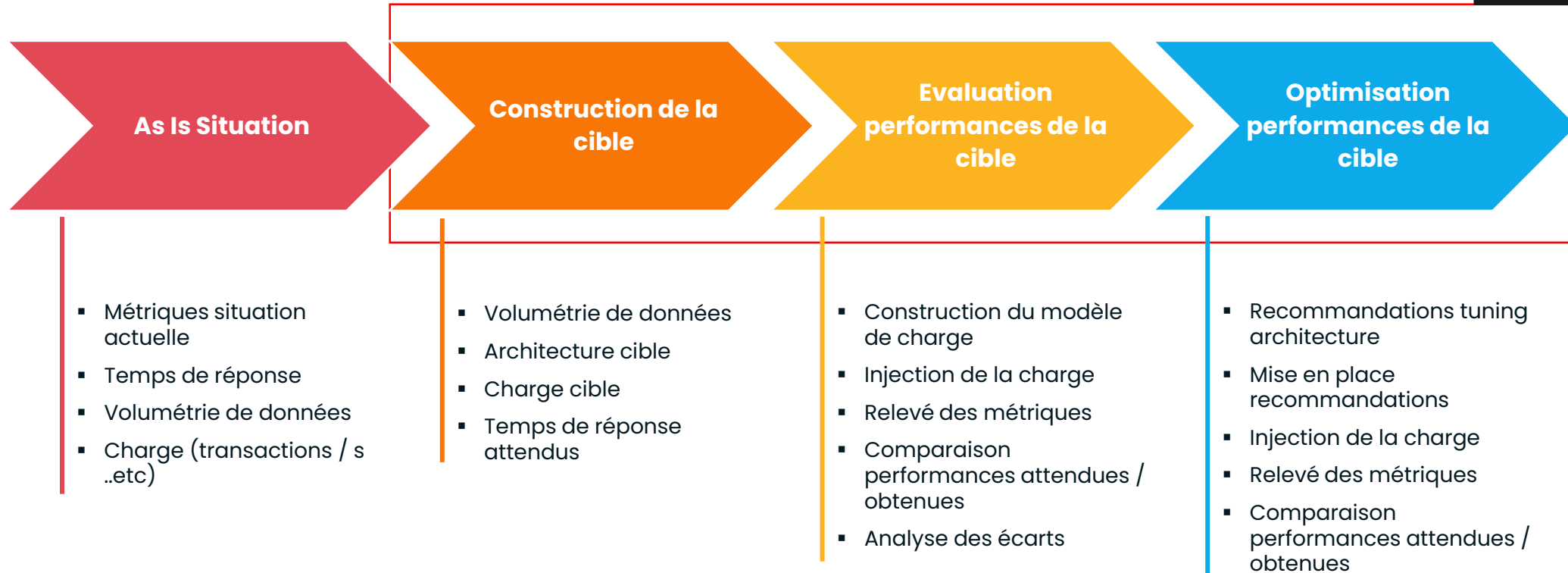
Test de performance et agilité



Test dans le sprint



Diagnostic des performances



1. Initialisation projet Gatling (IntelliJ)

Présentation Gatling

- Open source Français (Offre Entreprise payante disponible)
- Conçu pour les développeurs (Performance As a Code)
- Intégration simplifiée au CI/CD et gestionnaires de code.
- Support de plusieurs protocoles Web (via les implémentations en Java ou avec des plugins propriétaires)
- Support de plusieurs langages de programmation : Scala, Java, JavaScript/TypeScript, Kotlin
- Dispose d'un enregistreur de script
- Rapports natifs : Les rapports HTML complet et disponibles dans en Open Source.
- Injecteurs de charge on premise
- Intégration InfluxDb / Grafana
- Intégré aux cloud (AWS, GCP, Azure)

Comparatif Gatling / NeoLoad / Jmeter










Caractéristique	Gatling	JMeter	NeoLoad
Architecture	Basé sur Akka et Netty, modèle non-bloquant	Basé sur Java, utilise des threads Java	Architecture distribuée propriétaire
Script	Code Scala, approche DSL	Interface graphique + code XML/Java	Interface graphique + JavaScript
Évolutivité	Très bonne grâce au modèle d'acteurs	Limitée (1 thread par utilisateur virtuel)	Bonne avec distribution de charge
Courbe d'apprentissage	Moyenne à élevée (nécessite connaissance de Scala)	Faible à moyenne	Faible à moyenne
Rapports	Rapports HTML natifs détaillés	Basiques, extensions nécessaires	Rapports riches et personnalisables
Licence	Open Source (Apache 2.0)	Open Source (Apache 2.0)	Commercial
Coût	Gratuit (version entreprise disponible)	Gratuit	Payant
Interface	Code principalement, interface minimaliste	Interface graphique complète	Interface graphique riche
Intégration CI/CD	Excellente	Bonne	Bonne
Support	Communauté + support payant	Communauté	Support commercial
Maintenance des scripts	Facile (code structuré)	Plus complexe	Moyenne
Monitoring	Basique, intégrations possibles	Basique, plugins disponibles	Avancé, intégré



Installation Gatling

- Vérifier la version Java
- Télécharger Gatling (version Java) :
<https://gatling.io/products/download?hsCtaAttrib=174075950242>
- Dézipper le fichier gatling-maven-plugin-demo-java-main.zip
- Vérifier le contenu du dossier

```
C:\Apps\jdk-23.0.2\bin>java --version
openjdk 23.0.2 2025-01-21
OpenJDK Runtime Environment (build 23.0.2+7-58)
OpenJDK 64-Bit Server VM (build 23.0.2+7-58, mixed mode, sharing)
```

Nom		^
	.gatling	
	.github	
	.mvn	
	src	
	.gitignore	
	mvnw	
	mvnw.cmd	
	pom.xml	
	README.md	

Exécution du test










- Vérifier la version Java (java -version)
- Télécharger Gatling (version Java) :
<https://gatling.io/products/download?hsCtaAttrib=174075950242>
- Dézipper le fichier gatling-maven-plugin-demo-java-main.zip
- Gatling est fourni avec un test prêt à l'exécution : ComputerDatabase
- Vérifier le contenu du dossier

- Exécuter : **mvnw gatling:test**

./mvnw gatling:test -D"gatling.simulationClass= computerdatabase.ComputerDatabaseSimulation"

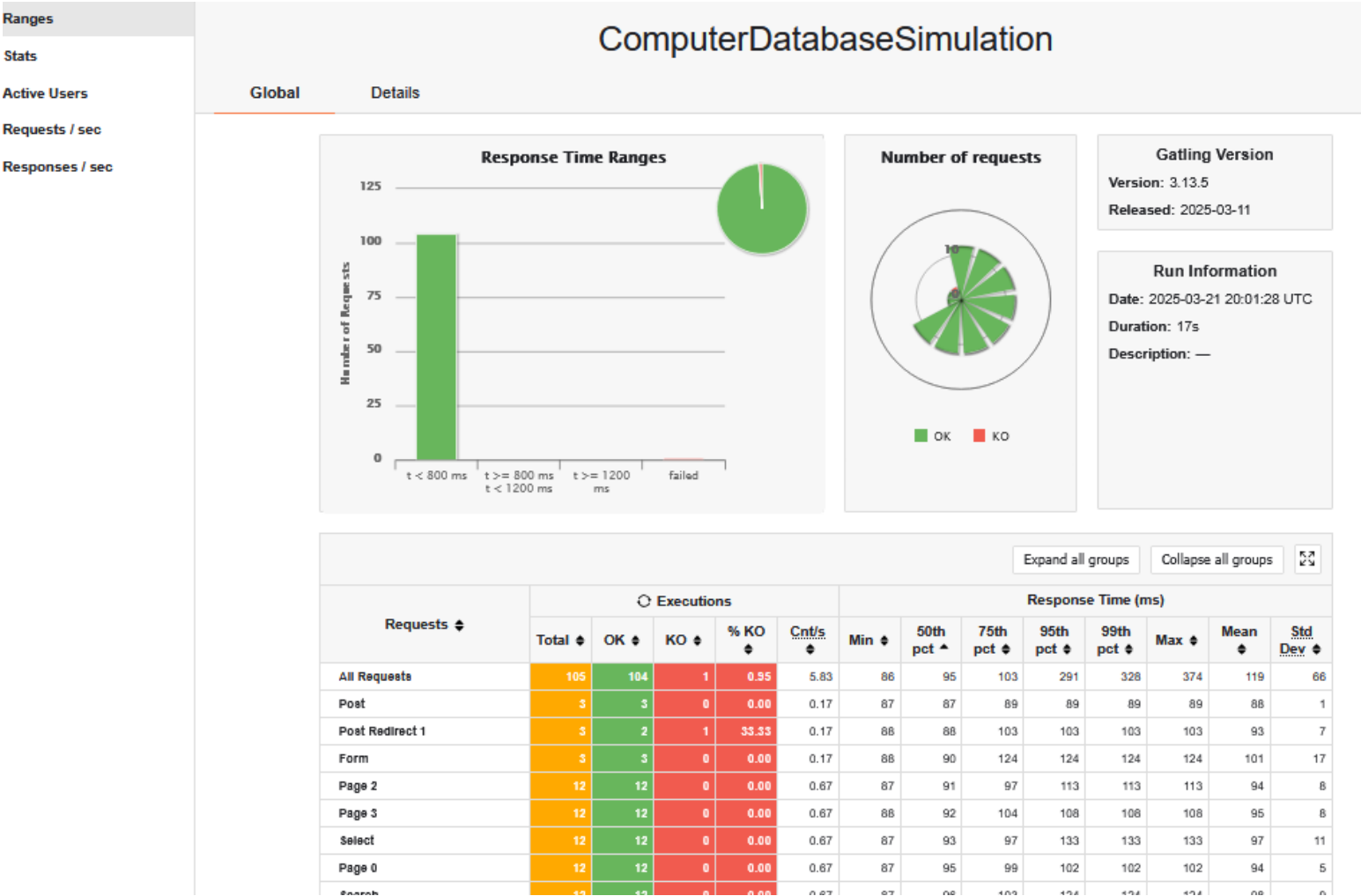
- Le résultat du test est disponible dans le dossier **target/gatling**

```
C:\Apps\jdk-23.0.2\bin>java --version
openjdk 23.0.2 2025-01-21
OpenJDK Runtime Environment (build 23.0.2+7-58)
OpenJDK 64-Bit Server VM (build 23.0.2+7-58, mixed mode, sharing)
```

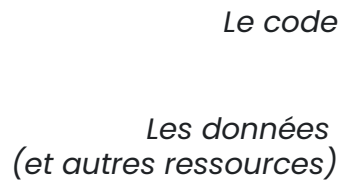
Nom		^
	.gatling	
	.github	
	.mvn	
	src	
	.gitignore	
	mvnw	
	mvnw.cmd	
	pom.xml	
	README.md	

Le rapport de test

- Gatling génère les données de test dans un fichier simulation.log
- A la fin de l'exécution, Gatling génère un rapport html complet.
- La version gratuite ne dispose pas de fonction de comparaison de rapports.



© 2005 Blackwell Publishing Ltd



Mode debug : 4 méthodes

1. Ajouter une configuration maven

Commande

gatling:test -Dgatling.sameProcess=true

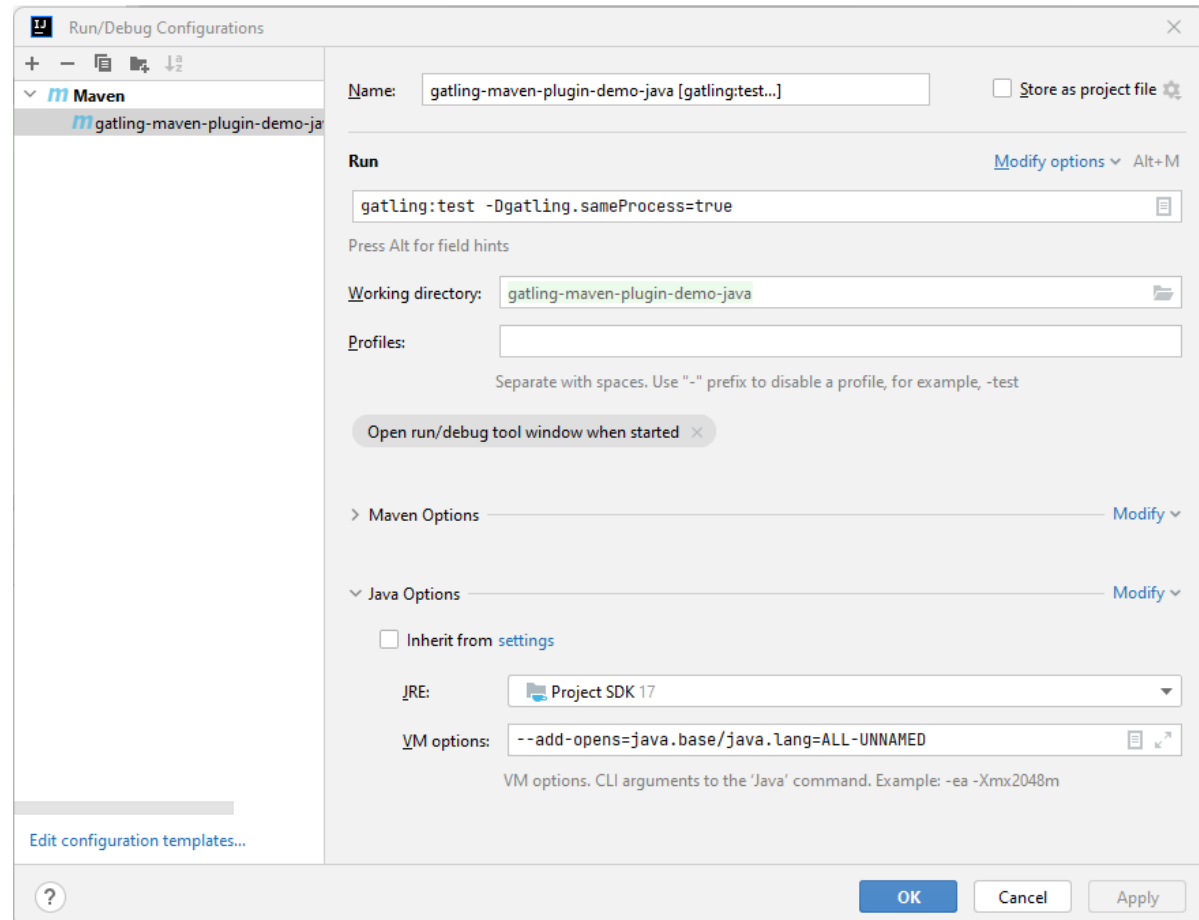
VM options

--add-opens=java.base/java.lang=ALL-UNNAMED

2. Modifier le fichier logback-test.xml
3. Ajouter la capture / affichage du body de la réponse

```
.check(bodyString().saveAs("body"))  
.exec(session -> {  
  System.out.println("075:PUT employee : " + session.getString("body"));  
  return session;  
})
```

4. Utiliser fiddler et configurer le proxy dans
HttpProtocolBuilder



La structure d'une simulation

Feeder (données de tests)

CSV, JSON, SiteMap

Array, List

JDBC, Redis ... etc

ChainBuilder

Séquence de requêtes http

Rendre le test modulaire avec des transactions réutilisables

HttpProtocolBuilder

Paramètres communs pour toutes les requêtes http (base url, proxy, headers, browser agent)

ScenarioBuilder

Assemblage des ChainBuilder pour construire un ou plusieurs scénarios utilisateur (Virtual User)

Setup

Répartition des ScenarioBuilder et définition du modèle de charge

```
public class ComputerDatabaseSimulation extends Simulation {

    FeederBuilder<String> feeder = csv("search.csv").random();

    ChainBuilder search = exec(
        http("Home").get("/"), ... etc

    ChainBuilder browse = .... etc

    ChainBuilder edit =
        http("Form").get("/computers/new"),
        .exitHereIfFailed(); ...etc

    HttpProtocolBuilder httpProtocol =
        http.baseUrl("https://computer-database.gatling.io")
        .acceptHeader("text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8")

    ScenarioBuilder users = scenario("Users").exec(search, browse);
    ScenarioBuilder admins = scenario("Admins").exec(search, browse, edit);

    {
        setUp(
            users.injectOpen(rampUsers(10).during(10)),
            admins.injectOpen(rampUsers(2).during(10))
        ).protocols(httpProtocol);
    }
}
```


Enregistrer un virtual user

- ***mvnw gatling:recorder***
- 2 possibilités :
 1. En mode proxy : l'enregistrement des requêtes se fait en même temps que la réalisation des actions sur l'IHM.
 2. Avec un har : capture har (F12 → Network), et import du har dans le projet.

Gatling Recorder - Configuration

Gatling

Recorder mode: HTTP Proxy

Network

Listening port*: localhost HTTP/HTTPS 8000 HTTPS mode: Self-signed Certificate

Outgoing proxy: host: HTTP Username Password

Simulation Information

Package: io.gatling.demo Class Name*: RecordedSimulation Format*: Java 17

☒ Follow Redirects? ☒ Infer HTML resources? ☒ Automatic Referers? ☒ Remove cache headers?

☐ Use Class Name as request prefix? ☐ Use HTTP method and URI as request postfix? ☐ Save & check response bodies?

Output

Simulations folder*: C:\github_himhah-op\gatling-training\gatling-java\src\test\java Browse

Encoding: Unicode (UTF-8)

Filters

Java regular expressions that match the entire URI

AllowList

DenyList

*.js
*.css
*.gif
*.jpeg
*.jpg
*.ico

+ - Clear No static resources

Save preferences Start!

Exercice 1 : Enregistrement et rejeu



Ne jamais lancer un test de charge sur un site sans l'accord du propriétaire

- 0.1. Lancer ***mvnw gatling:recorder*** et démarrer l'enregistrement en mode proxy
- 0.2. Paramétrer le proxy et réaliser le test manuel sur orangehrm (<https://opensource-demo.orangehrmlive.com>)
Login → Création Employé → Saisie du détail → Recherche Employé par nom → Modification des coordonnées → Recherche Employé par matricule → Suppression de l'employé → Logout
- 0.4. Vérifier que les requêtes sont bien enregistrées.
- 0.5. Analyser le code généré et isoler le login et le logout dans un ChainBuilder.
- 0.6. Construire un ScenarioBuilder uniquement avec le login et le logout.
- 0.7. Exécuter le test (avec un seul utilisateur et sans supprimer les temps d'attente) et analyser les résultats et corriger les erreurs

Si l'application est en localhost, sur firefox positionner le paramètre `network.proxy.allow_hijacking_localhost` à **true (about:config)**

<https://learn.microsoft.com/en-us/microsoft-cloud/dev/dev-proxy/how-to/intercept-localhost-requests?pivots=client-operating-system-windows>

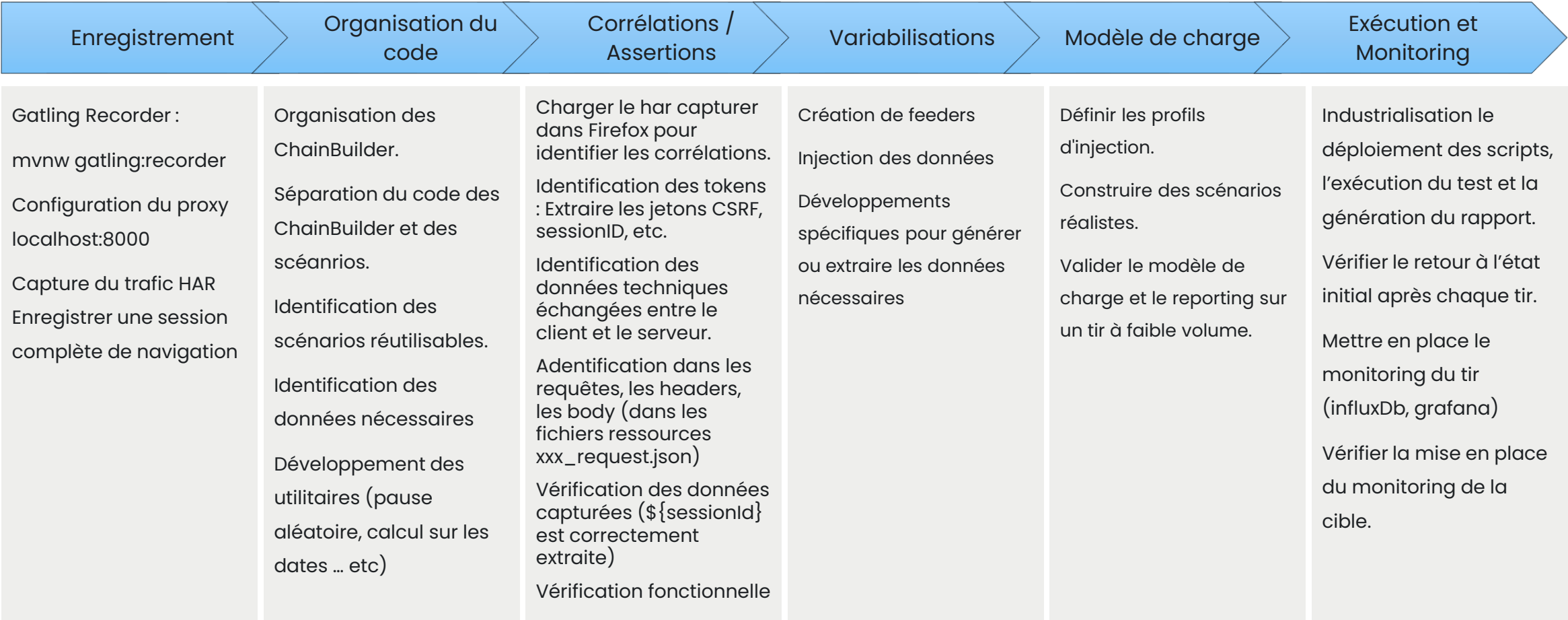
6. Développement Virtual User



Processus de développement

L'analyse préalable permet :

- la définition des objectifs (ex API de paiement 500 transactions/seconde temps de réponse < 200ms)
- l'identification des scénarios critiques (ex parcours d'achat complet, de la connexion à la confirmation de commande)
- La définition du modèle de charge : (Ex : pics attendus à 2000 utilisateurs simultanés pendant les soldes, objectif de temps de réponse moyen inférieur à 500ms)



Validation des scripts



Organisation du code

- **Supprimer tous les appels qui ne concernent pas l'application testée.**
- Remplacer RawRequestBody par ElRequestBody
- Séparer les actions dans les ChainBuilder des scénarios.
- Privilégier le format ChainBuilder `logout()` {
 return (... etc
- Caper les « pause » pour ne pas ralentir l'exécution pour la validation des scripts, ou les remplacer par une fonction de génération de pause aléatoire.
- Renommer les requêtes pour améliorer la lisibilité du code et du rapport.
- Supprimer les headers redondants.
- Supprimer les ressources statiques si nécessaires.

```
.exec(http("request_40")  
  .put("/web/index.php/api/v2/pim/employees/2/personal-details")  
  .headers(headers_40)  
  
  .body(RawRequestBody("fr/onepoint/training/addemployee/0040_request.json"))
```

```
ChainBuilder logout() {  
  return (  
    exec(http("request_83")  
      .get("/web/index.php/auth/logout")  
      .headers(headers_0)  
      .resources(  
        http("request_84")  
          .get("/web/index.php/core/i18n/messages")  
          .headers(headers_1)  
        )  
      )  
    )  
  .pause(ThinkTime())  
  );  
}
```

```
public static Duration thinkTime() {  
  Random rnd = new Random();  
  int thinkTimeMillis = rnd.nextInt(400) + 100;  
  return Duration.ofMillis(thinkTimeMillis);  
}
```

Corrélation et assertions

- Les corrélations sont réalisées avec la méthode check
- Les étapes à réaliser :
 - Identifier la donnée à corréler dans une requête, un header ou une ressource.
 - Dans le har rechercher la 1ere requête qui renvoie la donnée.
 - Capturer la donnée avec un check et vérifier que la capture est valide.
 - Remplacer la donnée par la variable capturée.

```
.exec(http("request_2")  
  .post("/web/index.php/auth/validate")  
  .headers(headers_2)  
  .formParam("_token",  
    "2d.QamKI6jipajHkDwVGLVje5X5KW4gxZt4uGciWLWs4QE.d8a-T5yw9t-  
u6UVsLYACNNyqfiVRtNcfIQB9G9blhTlx5r5xxNL0xYz2DA"))
```

Ce token est renvoyé par la requête /login

The screenshot shows the Burp Suite interface. On the left, a list of requests is displayed, with the request to localhost:8060/login selected. The right pane shows the HTML response of this request. The response contains a token value assigned to the 'token' variable in a JavaScript object, which is the data being correlated.

Domaine	Fichier	Taille
localhost:8060	/	3,46 Ko
localhost:8060	login	3,46 Ko
localhost:8060	chunk-vendors.css	249,97 Ko
localhost:8060	app.css?v=17213	163,76 Ko
localhost:8060	chunk-vendors.js	1,70 Mo
localhost:8060	app.js?v=172136	1,51 Mo
localhost:8060	messages	0 o
localhost:8060	favicon.ico?v=17	564 o
localhost:8060	ohrm_branding.p	30,41 Ko
localhost:8060	ohrm_logo.png	324,44 Ko
localhost:8060	blob.svg	289 o
localhost:8060	nunito-sans-v6-l	25,78 Ko
localhost:8060	bootstrap-icons.v	121,30 Ko
localhost:8060	nunito-sans-v6-l	25,82 Ko
localhost:8060	nunito-sans-v6-l	25,88 Ko

```
--oxd-secondary-four-alpha-50-color:rgba(118, 188, 33, 0.5);  
}  
<script>  
<strong>  
  We're sorry but orangehrm doesn't work properly without JavaScript enabled. Please enable it to co  
</strong>  
</script>  
<div id="app">  
  <h1>login  
    <div token="&quot;b611ea32ad3d8ff74b.8RXYsTRnKFzckG-r184exboyq4yLND5jcv0-JFL7F1o.pnyC2HA0SR5rwdX7iKEs  
    <div login-logo-src="&quot;/web/images/ohrm_logo.png&quot;">  
    <div login-banner-src="&quot;/web/images/ohrm_branding.png?v=1721362923870&quot;">  
    <div show-social-media="true">  
    <div is-demo-mode="false">  
    <div authenticators="[]">  
  </div>  
</div>  
<div plate v-slot:footer>  
  <div class="orangehrm-copyright-wrapper">  
    <div oxd-text tag="p" class="orangehrm-copyright">OrangeHRM OS 5.7</oxd-text>  
    <div tag="p" class="orangehrm-copyright">&copy; 2005 - 2025 <a href="http://www.orangehrm.com" target="_t  
  </div>  
</div>  
</div>  
</div>
```

Ajouter l'extraction du token dans la requête /login

```
.check(  
  regex(":token=|\"&(.*)&\"").saveAs("csrf_token"))
```

Puis remplacer la token (dans tous le projet) par la variable
`#{csrf_token}`

Vérifier la capture :

```
.exec ( session -> {  
  System.out.println(session.getString("csrf_token"));  
  return session;  
})
```

Exemple de check

Vérifications

Statut

```
.check(status().is(200))
.check(status().not(404))
.check(status().in(200, 201, 202))
```

Contenu

```
.check(bodyString().is("Contenu attendu"))
.check(bodyString().contains("utilisateur"))
```

Json

```
.check(jsonPath("$.name").is("John"))
.check(jsonPath("$.count").ofType(Integer.class).gt(5))
```

CSS

```
.check(css("#mainDiv").exists())
.check(css("h1").is("Titre principal"))
```

XPATH

```
.check(xpath("//button[@id='submit']").exists())
.check(xpath("//title").is("Ma Page"))
```

REGEX

```
.check(regex("tag-(\\w+)").count().is(5))
.check(regex("prix: (\\d+\\.\\d+)").ofType(Double.class).is(19.99))
```

Personnalisé

```
.check(bodyString().transform(body -> {
    // Logique personnalisée
    return body.contains("utilisateur") ? "OK" : "NOK";
}).is("OK"))
```

Extraction

Json

```
.check(jsonPath("$.users[*].name").findAll().saveAs("userNames"))
.check(jsonPath("$.id").saveAs("userId"))
```

CSS

```
.check(css("a.link").ofType(String.class).saveAs("liens"))
```

REGEX

```
.check(regex("id=([0-9]+)").saveAs("userId"))
.check(regex("<item>(.*?)</item>").findAll().saveAs("items"))
.check(regex("token=([a-zA-Z0-9]+)").saveAs("authToken"))
.check(regex("content: (.{10})").saveAs("partialContent"))
.check(regex("data=(.*?)&").saveAs("rawData"))
.check(regex("username=(?<name> \\w+)").ofType(String.class).saveAs("username"))
```

Personnalisé

```
.check(regex("date: (\\d{2}-\\d{2}-\\d{4})")
    .transform(date -> {
        // Convertir de DD-MM-YYYY à YYYY-MM-DD
        String[] parts = date.split("-");
        return parts[2] + "-" + parts[1] + "-" + parts[0];
    })
    .saveAs("formattedDate"))
```

Exercice 2 : Corrélation

- 0.1. Séparer les différentes actions du code dans des ChainBuilder.
- 0.2. Utiliser les ChainBuilder pour créer les parcours suivants :
 - Login → Consultation employé → Logout
 - Login → Ajout employé → Details → Contact → Logout
 - Login → Suppression employé → Logout
- 0.3. Identifier les données qui nécessitent une corrélation en utilisant la har enregistré.
- 0.4. Ajouter les « check » pour extraire les données et paramétrer les requêtes.
- 0.5. Exécuter le test au fur à mesure et ajouter les debugs des variables extraites.
- 0.6. Séparer le code des actions dans une classe dédiée.
- 0.7. Exécuter le test (avec un seul utilisateur et avec des pauses aléatoires) et analyser les résultats et corriger les erreurs.

3. Utilisation des données

feeders

Source de données (ex fichier CSV) pour injecter des valeurs dynamiques dans les scripts.

4 stratégies pour la lecture des feeders :

1. Queue : consomme les données dans l'ordre (par défaut)
2. Random : sélectionne les données aléatoirement
3. Circular : reprend au début quand toutes les données ont été utilisées
4. Shuffle : mélange les données avant utilisation (queue)

Types de feeders :

1. CSV (`csv("foo.csv")`) → séparateur « , », `tsv` → séparateur tabulation, `ssv` → séparateur « ; » ou `separatedValues("foo.txt", '#')`
2. JSON : `jsonFile("foo.json");`
3. JDBC feeder : `jdbcFeeder("databaseUrl", "username", "password", "SELECT * FROM users");`
4. Sitemap feeder : `sitemap("/path/to/sitemap/file");`
5. Redis feeder :
 - LPOP – Supprime et renvoie le premier élément de la liste
 - SPOP – Supprime et renvoie un élément aléatoire de l'ensemble
 - SRANDMEMBER – Renvoie un élément aléatoire de l'ensemble
 - RPOPLPUSH – Renvoie le dernier élément de la liste et l'enregistre comme premier élément d'une autre liste

Modes de chargement :

1. Egger : chargement en mémoire (pour des fichiers de petites tailles) : `csv("foo.csv").eager().random();`
2. batch : pour les gros fichiers (buffer par défaut 2000) : `csv("foo.csv").batch(200).random();`
3. Zip : pour les très gros fichiers : `csv("foo.csv.zip").unzip();`

Exercice 3 : feeders

0.1. Reprendre l'exercice 2 et remplacer les données de l'employé (nom, prénom, matricule) par un fichier csv

0.2. Le test de charge comporte 3 parcours différents :

Login → Consultation employé → Logout

Login → Ajout employé → Details → Contact → Logout

Login → Suppression employé → Logout

Proposer une méthode pour gérer les données des différents parcours

6. Modèle de montée en charge

Répartition de la charge

Gatling permet de définir la répartition de la charge sur les différents parcours :

1. randomSwitch : répartition aléatoire respectant un pourcentage par parcours
2. uniformRandomSwitch : répartition aléatoire sans définition préalable du pourcentage
3. roundRobinSwitch : exécuter différentes branches de scénario selon un ordre séquentiel prédéfini

```
.exec(http("050:GET api/pim/employees"))
```

```
.get("/web/index.php/api/v2/pim/employees?nameOrId=#{nom}&includeEmployees=onlyCurrent")  
.headers(headers_6)  
.pause(thinkTime())  
.exec(http("051:GET api/pim/employees"))
```

```
.get("/web/index.php/api/v2/pim/employees?limit=50&offset=0&model=detail&nameOrId=#{nom}&includeEmployees=onlyCurrent&sortField=employee.firstName&sortOrder=ASC")  
.headers(headers_6)  
.check(jsonPath("$.data[0].empNumber").saveAs("employeeId"))  
.exec(session -> {  
  System.out.println("051:GET api/pim/employees : employeeId = " + session.getString("employeeId"));  
  return session;  
})  
.pause(thinkTime())  
.... etc
```

```
public static ChainBuilder login()
```

```
public static ChainBuilder logout()
```

```
public static ChainBuilder add_employee()
```

```
public static ChainBuilder search_employee(String nom)
```

```
public static ChainBuilder employee_details()
```

```
public static ChainBuilder employee_contact()
```

```
public static ChainBuilder delete_employee()
```

```
public static ChainBuilder vu_add_employee() {
```

```
  return (  
    exec(login())  
    .exec(add_employee())  
    .exec(employee_details())  
    .exec(employee_contact())  
    .exec(logout())  
  );  
}
```

```
public static ChainBuilder vu_search_employee() {
```

```
  return (  
    exec(login())  
    .exec(search_employee())  
    .exec(logout())  
  );  
}
```

```
private final ScenarioBuilder scn = scenario("OrangeHRM")  
.during(Duration.ofMinutes(5)).on(  
  pace(Duration.ofSeconds(5))  
  .randomSwitch().on(  
    percent(10.0).then(exec(OrangeHRM.vu_add_employee())),  
    percent(90.0).then(exec(OrangeHRM.vu_search_employee()))  
  ));  
{  
  setUp(  
    scn.injectOpen(rampUsers(2).during(Duration.ofSeconds(10)))  
  ).protocols(httpProtocol);  
}
```

Modèles de charge

Ouvert

Les utilisateurs virtuels sont injectés selon un débit spécifique, indépendamment de la durée d'exécution de leurs scénarios.

- Les utilisateurs sont injectés à un rythme prédéfini (par exemple, 10 utilisateurs par seconde)
- Le nombre d'utilisateurs simultanés varie en fonction du temps de réponse du système
- Simule mieux le trafic réel du web où les utilisateurs arrivent indépendamment les uns des autres

- Exemples

```
scn.injectOpen(rampUsers(2).during(Duration.ofSeconds(10)))
```

```
scn2.injectOpen(constantUsersPerSec(2).during(Duration.ofSeconds(10)))
```

Fermé

Un nombre fixe d'utilisateurs virtuels est maintenu dans le système. Lorsqu'un utilisateur termine son scénario, il est immédiatement remplacé par un nouvel utilisateur.

- Maintient un nombre constant d'utilisateurs actifs
- Simule mieux les systèmes où le nombre d'utilisateurs est limité (comme une application interne d'entreprise)
- La charge dépend du temps de réponse du système

- Exemples

```
scn2.injectClosed(constantConcurrentUsers(10).during(Duration.ofSeconds(10)))
```

```
scn2.injectClosed(rampConcurrentUsers(10).to(100).during(Duration.ofSeconds(10)))
```

Exercice 4 : Modèle de charge

- 0.1. Reprendre l'exercice 3
- 0.2. Ajouter la répartition de charge suivante :
 - 80% Login → Consultation employé → Logout
 - 10% Login → Ajout employé → Details → Contact → Logout
 - 5% Login → Suppression employé → Logout
- 0.3 Définir un modèle de charge ouvert : montée en charge de 2 VU toutes les 30 secondes jusqu'à 10 VU. Maintenir la charge pendant 10 mn
- 0.4 Pour lancer ce tir il est nécessaire de déployer l'application OrangeHRM localement (ou sur un serveur dédié)
Docker : **orangehrm/orangehrm:latest** et **mysql:8.4.4**

6. Cas pratique

Gestion de projets Agile avec Taiga.io



Cas pratique

Installer taiga.io dur docker (en local ou sur un serveur commun) :

```
git clone https://github.com/taigaio/taiga-docker.git
cd taiga-docker
git checkout stable
./launch-taiga.sh
/taiga-manage.sh createsuperuser
```

1. Créer un projet Scrum (vérifier que le module Issue est actif : Settings → Project → Modules)
2. Avec Gatling capturer le parcours : Login → Issues List → Create Issue (Subject / Description / Type / Severity / Priority) → Update Issue (Status, Priority) → Search Issue → Delete Issue → Logout
3. Définir 4 parcours :
 - a. Login → Issues List → Search Issue → Logout
 - b. Login → Issues List → Issues List → Create Issue (Subject / Description / Type / Severity / Priority) → Logout
 - c. Login → Issues List → → Update Issue (Status, Priority) → Logout
 - d. Login → Issues List → Search Issue → Delete Issue → Logout
4. Répartir la charge comme suit : a : 75% - b : 15% - c : 10% - d : 10%
5. Réaliser les tirs de performance et identifier les limites de performance de l'application.



Merci

