

# Simulateur de tests de charge et performance dans une architecture n-tiers

10 juin 2024

Jaloulli Ibrahim  
Hannachi Youssef  
Chtioui Ziad  
Jouanneau Nicolas

Rapport Projet Master 1 Classique  
Master 1 ILSEN-SICOM

**Responsables**  
Yezekael Hayel  
Emmanuel Puy-Cazcarra

## Sommaire

<b>Titre</b>	<b>1</b>
<b>Sommaire</b>	<b>2</b>
<b>1 Présentation Générale du Projet</b>	<b>4</b>
1.1 Contexte et Problématique . . . . .	4
1.1.1 Introduction au projet . . . . .	4
1.1.2 Contexte industriel et académique . . . . .	4
1.1.3 Problématique et enjeux . . . . .	4
1.2 Objectifs du Projet . . . . .	5
1.2.1 Objectifs généraux . . . . .	5
1.2.2 Objectifs spécifiques . . . . .	5
1.3 Organisation du Rapport . . . . .	6
1.3.1 Structure du rapport . . . . .	6
1.3.2 Méthodologie de travail . . . . .	7
<b>2 Phase d'Analyse (1er Semestre)</b>	<b>8</b>
2.1 Analyse des Besoins . . . . .	8
2.1.1 Besoins du site E-commerce . . . . .	8
Performance et Scalabilité . . . . .	8
Sécurité . . . . .	9
Convivialité et Accessibilité . . . . .	9
Maintenance et Extensibilité . . . . .	9
2.1.2 Besoins de la solution de simulation . . . . .	9
Performance et Scalabilité . . . . .	9
Convivialité et Accessibilité . . . . .	10
Maintenance et Extensibilité . . . . .	10
2.2 Étude de l'Architecture . . . . .	10
2.2.1 Conception de l'architecture N-tiers pour le site e-commerce . . . . .	10
2.2.2 Analyse des composants . . . . .	11
2.3 Choix des outils technologiques technologique . . . . .	11
2.3.1 Développement front-End . . . . .	11
2.3.2 Développement back-End . . . . .	12
2.3.3 Développement de la solution de simulation . . . . .	13
2.3.4 Outils de Collaboration et de Planification . . . . .	13
2.4 Choix des Outils de Test et de Monitoring . . . . .	15
2.4.1 Évaluation des Outils de Test de Charge . . . . .	15
2.4.2 Évaluation des Outils de Monitoring . . . . .	16
2.4.3 Justification des choix effectués . . . . .	17
2.5 Études Choix des Différents Types de Test . . . . .	18
2.6 Modélisation des scénarios de test . . . . .	19
2.6.1 Définition des scénarios de test . . . . .	19
2.6.2 Identification des indicateurs de performance . . . . .	19
<b>3 Phase de Réalisation (2ème Semestre)</b>	<b>21</b>
3.1 Développement . . . . .	21
3.1.1 Mise en place de l'infrastructure avec les VMs . . . . .	21
3.1.2 Conception et Développement du site e-commerce . . . . .	22
3.1.3 Création et intégration de la base de données . . . . .	26

3.2	Mise en Œuvre des Outils de Test et de Monitoring . . . . .	27
3.2.1	Configuration de JMeter pour les tests de charge . . . . .	27
3.2.2	Intégration de Grafana pour le monitoring des performances . . . . .	27
3.3	Exécution des Tests de Charge . . . . .	29
3.3.1	Déroulement des tests de charge . . . . .	29
3.3.2	Collecte et analyse des résultats . . . . .	31
3.3.3	Ajustements et optimisation . . . . .	31
3.4	Analyse des Résultats . . . . .	32
3.4.1	Présentation des résultats des tests . . . . .	32
3.4.2	Analyse des performances . . . . .	32
3.4.3	Identification des goulots d'étranglement et propositions d'améliorations	33
<b>4</b>	<b>Difficultés Rencontrées</b>	<b>34</b>
4.1	Méconnaissance des logiciels et leur fonctionnement . . . . .	34
4.2	Lien entre Grafana et Prometheus . . . . .	34
4.3	Connexion entre les différentes VM . . . . .	34
4.4	Problèmes pour l'app web entre le front et le back . . . . .	34
<b>5</b>	<b>Conclusion et Perspectives</b>	<b>36</b>
5.1	Résumé des travaux réalisés . . . . .	36
5.2	Bilan par rapport aux objectifs initiaux . . . . .	36
5.3	Perspectives d'amélioration et de prolongement du projet . . . . .	36
<b>6</b>	<b>Bibliographie</b>	<b>37</b>

## 1 Présentation Générale du Projet

### 1.1 Contexte et Problématique

#### 1.1.1 Introduction au projet

Avec l'essor du commerce électronique, les entreprises sont de plus en plus dépendantes de la performance et de la fiabilité de leurs systèmes d'information (SI). La capacité à fournir une expérience utilisateur fluide et rapide est cruciale pour maintenir la satisfaction des clients et assurer la compétitivité sur le marché. Dans ce contexte, ce projet vise à développer une plateforme de simulation de tests de performance pour une architecture N-Tiers, typique des sites e-commerce. Cette plateforme permettra de simuler des scénarios de tests, de surveiller les performances en temps réel et d'identifier les éventuels goulots d'étranglement.

#### 1.1.2 Contexte industriel et académique

Les systèmes d'information modernes, en particulier ceux utilisés par les sites e-commerce, sont souvent basés sur une architecture N-Tiers. Cette architecture sépare les différentes fonctions de l'application (présentation, logique d'application et données) en couches distinctes, ce qui améliore la modularité et la scalabilité du système. Cependant, cette complexité accrue nécessite des outils et des méthodes sophistiqués pour garantir que chaque couche fonctionne de manière optimale et que l'ensemble du système répond aux exigences de performance et de disponibilité.

Dans le contexte académique, de nombreux chercheurs se concentrent sur l'optimisation des performances des systèmes distribués et sur le développement de nouvelles techniques de monitoring et de test. Ces recherches fournissent des bases théoriques solides et des innovations technologiques qui peuvent être appliquées dans un contexte industriel pour améliorer la fiabilité et la performance des SI.

#### 1.1.3 Problématique et enjeux

La principale problématique pour les entreprises opérant des sites e-commerce est de garantir la performance et la disponibilité continue de leur SI, surtout en période de forte demande, comme lors des promotions ou des périodes de fêtes. Les enjeux sont multiples :

**Visibilité et monitoring** : Il est essentiel de disposer d'une visibilité en temps réel sur le fonctionnement du SI pour détecter rapidement les anomalies et les défaillances potentielles.

**Performance technique et applicative** : Les tests de performance doivent couvrir à la fois les aspects techniques (réponse des serveurs, temps de chargement des pages, etc.) et applicatifs (logique métier, interactions utilisateur, etc.).

**Scalabilité** : Le système doit pouvoir s'adapter à des variations de charge importantes sans dégradation des performances.

**Expérience utilisateur** : Une mauvaise performance du site peut entraîner une baisse de la satisfaction des clients, voire des pertes financières directes dues à des abandons de panier ou à une diminution des ventes.

**Optimisation des ressources** : Une bonne gestion des performances permet d'optimiser l'utilisation des ressources (serveurs, réseaux, bases de données), réduisant ainsi les coûts opérationnels.

Pour répondre à ces enjeux, les entreprises doivent mettre en place des solutions de monitoring et de test de performance dès la phase de conception du système et tout au

long de son cycle de vie. L'objectif de ce projet est de proposer une plateforme capable de simuler des scénarios de tests réalistes et d'évaluer les performances d'une architecture N-Tiers, afin de garantir la qualité et la performance du SI en production.

## 1.2 Objectifs du Projet

### 1.2.1 Objectifs généraux

Le projet vise avant tout à concevoir et mettre en œuvre une plateforme de simulation de tests de performance dédiée à une architecture N-Tiers, couramment utilisée pour les sites e-commerce. Cette plateforme doit répondre aux exigences croissantes des entreprises en matière de surveillance et d'optimisation des performances de leur système d'information (SI). En termes concrets, cela signifie permettre une évaluation approfondie des performances techniques et applicatives du SI dans diverses conditions de charge, offrir la capacité de simuler des scénarios d'utilisation réalistes, et assurer une surveillance continue des performances en temps réel. L'objectif est également d'identifier rapidement les anomalies et les défaillances, de faciliter l'optimisation des ressources et de générer des rapports détaillés pour une analyse approfondie.

### 1.2.2 Objectifs spécifiques

#### Phase 1 - Étude

La première phase se concentrera sur la définition de l'expression des besoins, impliquant une identification précise des exigences des parties prenantes concernant la performance du SI. Parallèlement, une analyse approfondie des différentes solutions de simulation de tests de performance disponibles sur le marché sera effectuée, en évaluant leurs fonctionnalités, leurs avantages et leurs inconvénients.

#### Phase 2 - Maquette

Dans la phase de maquette, un environnement de test complet sera préparé, incluant la mise en place de l'infrastructure réseau, système et applicative nécessaire pour simuler un environnement de production. Cette phase comprendra également la sélection et la qualification de la solution de simulation de tests de performance présélectionnée, afin de s'assurer de sa pertinence et de son efficacité dans l'environnement de test mis en place.

#### Phase 3 - Monitoring

La troisième phase portera sur le développement des outils de monitoring. Des tableaux de bord interactifs seront conçus et développés pour permettre un suivi en temps réel des tests de performance et évaluer la qualité de service délivrée par le SI. Les outils de monitoring seront intégrés pour collecter et analyser les données de performance pendant les tests, offrant ainsi une visibilité continue sur le fonctionnement du système.

#### Phase 4 - Tests

La phase finale consistera en la conception et la réalisation des scénarios de test. Des scénarios de tests détaillés seront développés pour représenter des situations d'utilisation typiques et des charges de travail variées. Ces tests seront exécutés dans des conditions quasi-réelles afin d'évaluer les performances techniques et applicatives du SI. Les résultats des tests seront analysés en profondeur pour identifier les points faibles, les goulots d'étranglement et proposer des améliorations concrètes.

#### Livrables attendus

À la fin de ce projet, plusieurs livrables sont attendus. Un rapport détaillant les besoins

fonctionnels et non fonctionnels identifiés lors de la phase d'étude sera produit, ainsi qu'une analyse comparative des solutions de simulation de tests de performance étudiées, avec une recommandation sur la solution la plus appropriée. Un environnement de test configuré, comprenant l'infrastructure réseau, système et applicative, sera mis en place. Des tableaux de bord de monitoring interactifs seront développés pour le suivi en temps réel des performances. Enfin, des scénarios de tests documentés et des rapports détaillant les résultats des tests de performance et les recommandations pour les améliorations seront fournis. En atteignant ces objectifs, le projet fournira aux entreprises un outil puissant pour assurer et optimiser les performances de leurs systèmes d'information en architecture N-Tiers, garantissant ainsi une meilleure qualité de service et une satisfaction accrue des utilisateurs finaux.

## 1.3 Organisation du Rapport

### 1.3.1 Structure du rapport

Le rapport est structuré de manière à refléter les différentes phases du projet et à fournir une documentation claire et exhaustive de chaque étape. Il est divisé en deux grandes parties correspondant aux deux semestres du projet, chacune subdivisée en sections détaillées.

#### Phase d'Analyse (1er Semestre) :

##### Analyse des Besoins :

Recueil des besoins fonctionnels et non fonctionnels auprès des parties prenantes. Identification des critères de performance et des attentes spécifiques en termes de disponibilité, de scalabilité et de sécurité.

##### Étude de l'Architecture :

Conception de l'architecture N-tiers pour le site e-commerce, incluant la séparation des couches de présentation, de logique métier et de données. Choix des composants technologiques (serveur web, base de données, etc.) et justification de ces choix en fonction des besoins identifiés.

##### Choix des Outils Technologiques :

Sélection des frameworks, langages de programmation, bases de données, et autres technologies nécessaires pour le développement du site e-commerce et de l'infrastructure de test.

##### Choix des Outils de Test et de Monitoring :

Évaluation et sélection des outils de test de charge (comme JMeter, LoadRunner) et des outils de monitoring (comme Grafana, Kibana). Justification des choix effectués en fonction de leur pertinence, de leur performance et de leur coût.

##### Études et Choix des Différents Types de Test :

Identification des types de tests à réaliser (tests de charge, tests de stress, tests de résistance, etc.). Sélection des tests pertinents pour évaluer les différents aspects de la performance du système.

##### Modélisation des Scénarios de Test :

Définition des scénarios de test basés sur les cas d'utilisation réels et les exigences de performance. Identification des indicateurs de performance clés (KPI) et des métriques

pertinentes pour chaque scénario.

#### **Phase de Réalisation (2ème Semestre) :**

##### **Développement :**

Mise en place de l'infrastructure avec des machines virtuelles (VMs) pour répliquer l'architecture N-tiers. Développement du site e-commerce, incluant le front-end (interface utilisateur) et le back-end (logique métier et gestion des données). Création et intégration de la base de données pour gérer les transactions et les données des utilisateurs.

##### **Mise en Œuvre des Outils de Test et de Monitoring :**

Configuration de JMeter pour la création et l'exécution des tests de charge. Intégration de Grafana et d'autres outils de monitoring pour la visualisation des performances en temps réel.

##### **Exécution des Tests de Charge :**

Planification et déroulement des tests de charge en suivant les scénarios de test définis. Collecte des données de performance pendant les tests et analyse des résultats pour identifier les goulots d'étranglement et les points faibles. Ajustements et optimisation de l'architecture et des composants pour améliorer les performances.

##### **Analyse des Résultats :**

Présentation des résultats des tests sous forme de rapports détaillés. Analyse des performances globales du système et identification des améliorations nécessaires. Proposition de solutions pour résoudre les problèmes de performance identifiés.

##### **Conclusion et Perspectives :**

Résumé des travaux réalisés au cours du projet. Évaluation du projet par rapport aux objectifs initiaux et analyse des résultats obtenus. Discussion des perspectives d'amélioration et des possibilités de prolongement du projet, incluant des recommandations pour des travaux futurs.

### **1.3.2 Méthodologie de travail**

Le projet sera mené selon une méthodologie agile, permettant des ajustements itératifs et une amélioration continue. Les principales étapes méthodologiques incluent :

**Réunion :** Des réunions ont été faites de manière régulière afin de s'assurer que chaque membre du groupe arrive à avancer en concordance avec le planning

**Conception et Prototypage :** Développement d'une maquette de l'environnement de test et des premiers prototypes des scénarios de test. Cette phase permet de valider les choix technologiques et d'ajuster les configurations avant la mise en œuvre complète.

**Développement Itératif :** Mise en place progressive des outils et des scénarios de test, avec des cycles de développement courts et des feedbacks réguliers. Chaque itération inclut des phases de planification, de développement, de test et de revue.

**Analyse des Données :** Collecte systématique des données de performance pendant les tests. Analyse approfondie des résultats pour identifier les tendances, les anomalies, et les points d'amélioration. Les résultats de chaque test sont documentés et utilisés pour guider les ajustements suivants.

## 2 Phase d'Analyse (1er Semestre)

### 2.1 Analyse des Besoins

Pour garantir la validité et la pertinence de nos tests, il est crucial de définir clairement les besoins fonctionnels et non fonctionnels de notre site E-commerce ainsi que ceux de l'ensemble de notre solution de simulation et de visualisation des tests de performance. En définissant ces besoins, nous nous assurons que notre site E-commerce est représentatif d'un cas réel et que notre solution globale répond aux objectifs fixés.

#### 2.1.1 Besoins du site E-commerce

Besoins fonctionnels

- **Gestion des utilisateurs** : Le site doit permettre aux utilisateurs de créer un compte et de se connecter. La gestion des utilisateurs comprend l'inscription et la connexion sécurisée, ce qui permet aux utilisateurs d'avoir un accès personnalisé à leurs informations de compte et de bénéficier d'une expérience d'achat plus fluide et sécurisée. Les utilisateurs enregistrés peuvent retrouver plus facilement leurs paniers, ce qui améliore leur expérience globale sur le site. En outre, les utilisateurs doivent pouvoir accéder à leur profil pour changer leurs informations personnelles, telles que leur mot de passe.
- **Consulter les produits** : Les utilisateurs doivent pouvoir naviguer et consulter l'ensemble des produits disponibles sur le site. Chaque produit doit être présenté avec des informations détaillées telles que le nom, la description, le prix, et les images. L'objectif est de fournir une interface utilisateur claire et intuitive qui permet aux visiteurs de parcourir facilement le catalogue de produits.
- **Recherche et filtrage des produits** : Le catalogue de produits doit inclure une fonctionnalité de recherche avancée permettant aux utilisateurs de trouver rapidement les produits qu'ils recherchent. Les utilisateurs doivent pouvoir filtrer les produits par catégories, telles que vêtements, accessoire, etc., et par fourchettes de prix. Cela permet une expérience utilisateur plus personnalisée et efficace, rendant la navigation sur le site plus agréable et ciblée.
- **Gestion de panier** : Chaque utilisateur doit avoir un panier d'achat personnel où il peut ajouter ou supprimer des articles. Le panier doit conserver les produits sélectionnés par l'utilisateur jusqu'à ce qu'il soit prêt à passer à la caisse. De plus, des options pour sauvegarder le panier pour plus tard doivent être disponibles. L'affichage du total des articles dans le panier, y compris les frais de livraison, doit être inclus pour offrir une transparence totale sur le coût final.

Besoins non fonctionnels

#### Performance et Scalabilité

- **Temps de réponse** : Le site doit offrir des temps de réponse rapides pour toutes les actions utilisateur, telles que le chargement des pages, la recherche de produits, et l'ajout d'articles au panier. Idéalement, les pages doivent se charger en moins de 2 secondes afin d'offrir une expérience utilisateur fluide.
- **Scalabilité** : Le site doit être capable de gérer une augmentation du nombre d'utilisateurs et de transactions sans perte de performance. Cela inclut la capacité à ajouter de nouvelles fonctionnalités ou à augmenter la capacité du serveur en fonction de la demande.

## Sécurité

- **Protection des données :** Toutes les données des utilisateurs, y compris les informations personnelles et les données de paiement, doivent être protégées contre les accès non autorisés par des mesures de sécurité robustes, telles que le chiffrement SSL/TLS. De plus, les mots de passe des utilisateurs dans la base de données doivent être chiffrés de manière sécurisée, en utilisant des algorithmes de hachage robustes comme bcrypt ou Argon2, pour garantir leur protection contre toute compromission.
- **Authentification et Autorisation :** Le site doit garantir que seuls les utilisateurs autorisés puissent accéder aux zones sensibles. L'utilisation de protocoles sécurisés pour l'authentification et l'autorisation est essentielle.

## Convivialité et Accessibilité

- **Interface utilisateur intuitive :** Le site doit avoir une interface utilisateur intuitive et facile à naviguer pour tous les utilisateurs, indépendamment de leur niveau de compétence technique.
- **Compatibilité multi-plateforme :** Le site doit être compatible avec différents navigateurs web et dispositifs, y compris les ordinateurs de bureau, les tablettes, et les smartphones.
- **Accessibilité :** Le site doit respecter les normes d'accessibilité pour être utilisable par des personnes ayant des handicaps, par exemple en suivant les directives WCAG (Web Content Accessibility Guidelines).

## Maintenance et Extensibilité

- **Facilité de maintenance :** Le code du site doit être bien structuré et documenté pour faciliter la maintenance et les mises à jour futures.
- **Extensibilité :** Le site doit être conçu de manière modulaire pour permettre l'ajout de nouvelles fonctionnalités ou la modification des fonctionnalités existantes sans perturber les opérations en cours.

### 2.1.2 Besoins de la solution de simulation

#### Besoins fonctionnels

- **Simulation de charge :** La solution doit être capable de simuler différentes charges d'utilisateurs sur le site E-commerce pour tester ses performances. Cela inclut la génération de scénarios de test réalistes qui imitent les comportements des utilisateurs, tels que la navigation, la recherche de produits, et les achats.
- **Collecte des données de performance :** La solution doit pouvoir collecter des données détaillées sur les performances du site sous charge. Cela comprend le suivi des temps de réponse, les taux de réussite des transactions, les erreurs, et d'autres métriques pertinentes.
- **Visualisation des résultats :** Il est essentiel que les résultats des tests de performance soient présentés de manière claire et compréhensible. La solution doit inclure des outils de visualisation tels que des graphiques, des tableaux et des rapports qui permettent d'analyser facilement les données collectées.
- **Personnalisation des tests :** Les utilisateurs de la solution doivent pouvoir personnaliser les scénarios de test en fonction de leurs besoins spécifiques. Cela inclut la possibilité de définir différents types de charges, de simuler des pics de trafic, et de tester différentes configurations de serveurs.

#### Besoins non fonctionnels

#### Performance et Scalabilité

- **Efficacité des tests :** La solution de simulation doit être capable d'exécuter des tests de performance rapidement et efficacement, en minimisant l'impact sur les ressources du système testé.

- **Scalabilité :** La solution doit pouvoir s'adapter à des charges de test de plus en plus élevées. Elle doit supporter des simulations avec des milliers d'utilisateurs simultanés sans perte de précision ou de performance.

### Convivialité et Accessibilité

- **Interface utilisateur conviviale :** La solution doit offrir une interface utilisateur intuitive et facile à utiliser, même pour les utilisateurs non techniques.
- **Compatibilité multi-plateforme :** La solution doit être compatible avec différents systèmes d'exploitation et navigateurs web pour permettre une utilisation flexible.

### Maintenance et Extensibilité

- **Facilité de mise à jour :** La solution doit être conçue pour permettre des mises à jour régulières et faciles, incluant de nouvelles fonctionnalités ou des améliorations des fonctionnalités existantes.
- **Modularité et extensibilité :** La solution doit être modulaire et extensible, permettant l'ajout de nouveaux types de tests ou de nouvelles métriques de performance sans nécessiter des modifications majeures du système. De plus, elle doit être capable de s'appliquer à n'importe quelle architecture N-tiers, assurant ainsi une flexibilité et une adaptabilité à différentes configurations de systèmes.

## 2.2 Étude de l'Architecture

Dans cette partie, nous allons mettre en lumière notre architecture, qui représente une partie cruciale de notre étude. Cette dernière est définie en deux points majeurs : L'architecture N-tiers pour le site e-commerce sur lequel nous allons réaliser nos tests, et l'identification des composants nécessaires pour la mise en place de notre infrastructure afin de simuler un environnement de travail réel.

### 2.2.1 Conception de l'architecture N-tiers pour le site e-commerce

Comme indiqué précédemment, nous visons à développer une architecture N-tiers, plus précisément pour un site e-commerce. Pour cela, nous avons identifié trois tiers importants : la couche présentation, la couche métier et la couche données.

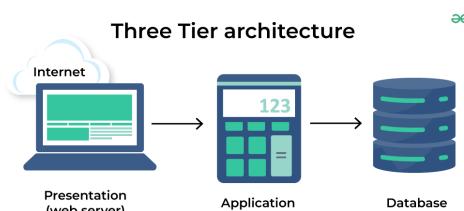


Figure 1. Architecture N-tiers

#### Couche Présentation

Responsable de l'interface utilisateur, permettant aux clients d'interagir avec le site e-commerce. Elle comprend les pages web, les formulaires et les éléments graphiques qui facilitent la navigation et l'achat en ligne. Cette partie est souvent développée en utilisant des technologies telles que HTML, CSS et JavaScript, ainsi que des frameworks comme React ou Angular.

#### Couche métier

Gère la logique de l'application. Elle traite les données reçues de la couche présentation, applique les règles métiers et exécute les opérations nécessaires. Cette couche est cruciale pour maintenir la cohérence des transactions. Des frameworks back-end tels que

Node.js, Django ou Spring peuvent être utilisés pour développer cette partie de l'architecture.

### Couche de Données

Est chargée du stockage, de la gestion et de la récupération des données. Elle inclut les bases de données et les systèmes de gestion de données qui conservent les informations sur les produits, les utilisateurs et les transactions. Cette couche assure la persistance et l'intégrité des données essentielles pour le fonctionnement du site e-commerce.

En séparant ces responsabilités en trois couches distinctes, nous pouvons développer une architecture modulaire et évolutive, facilitant la maintenance et l'amélioration continue du site e-commerce.

### 2.2.2 Analyse des composants

Les composants spécifiques de l'architecture comprennent :

**Serveur Web :** Apache ou Nginx sont utilisés pour gérer les requêtes HTTP et servir les contenus statiques et dynamiques.

**Base de données :** MySQL ou PostgreSQL sont sélectionnés pour leur fiabilité et leur performance dans la gestion des données transactionnelles.

**Cache :** Redis ou Memcached sont utilisés pour améliorer la rapidité d'accès aux données fréquemment utilisées, réduisant ainsi la charge sur la base de données principale.

**Jmeter :** Jmeter est mis en place pour effectuer les scénarios qui permettront de tester notre architecture sous une charge élevée.

## 2.3 Choix des outils technologiques technologique

### 2.3.1 Développement front-End

Le Front-End est la partie visible de l'application. C'est l'ensemble des éléments directement accessible par les utilisateurs. Dans notre cas, on a développé la partie frontale avec Angular.

- **Angular :** Développé par Google, Angular est un Framework open source écrit en JavaScript qui permet la création d'applications Web et plus particulièrement de ce qu'on appelle des SPA (Single Page Applications) : des applications web accessibles via une page web unique qui permet de fluidifier l'expérience utilisateur et d'éviter les chargements de pages à chaque nouvelle action. Le Framework est basé sur une architecture du type MVC et permet donc de séparer les données, le visuel et les actions pour une meilleure gestion des responsabilités.[marc2019angular]



Figure 2. Logo d'Angular

- **Angular Material :** Angular Material est une bibliothèque de composants d'interface utilisateur pour les développeurs Angular. Les composants d'interface utilisateur réutilisables d'Angular Material aident à créer des pages Web et des applications Web attrayantes, cohérentes et fonctionnelles.

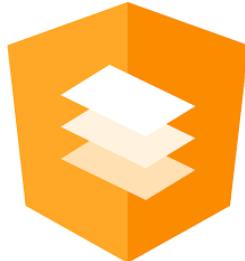


Figure 3. Angular Material

### 2.3.2 Développement back-End

Le développement back-end est le travail qui consiste à faire communiquer le serveur, le site ou l'application et la base de données. Il est développé avec Spring boot dans notre cas.

- **Spring Boot :** Spring Boot est un framework de développement JAVA. C'est une déclinaison du framework classique de Spring qui permet essentiellement de réaliser des microservices (ce sont la majeure partie du temps des services web qui sont regroupés en API).[\[axopen2018springboot\]](#)



Figure 4. Logo Spring Boot

- **Spring Security :** Spring Security est un Framework de sécurité léger qui fournit une authentification et un support d'autorisation afin de sécuriser les applications Spring. Il est livré avec des implémentations d'algorithme de sécurité populaires.[\[\[tamto2018springsecurity\]\]](#)



Figure 5. Logo Spring Security

- **MySQL :** MySQL est un système de gestion de bases de données relationnelles utilisant le langage de programmation SQL. Il propose une version open source qui permet à l'utilisateur d'accéder au code source et de le modifier, et une version entreprise permettant un accès aux dernières fonctionnalités du logiciel et au support fourni par Oracle, propriétaire et développeur actuel de MySQL. [\[\[redac2022mysql\]\]](#)



Figure 6. Logo MySql

### 2.3.3 Développement de la solution de simulation

Pour la solution de simulation, nous avons utilisé plusieurs outils technologiques :

- **VirtualBox** : VirtualBox est un logiciel de virtualisation qui permet de créer et de gérer des machines virtuelles. Il est open source et compatible avec de nombreux systèmes d'exploitation, facilitant ainsi le développement et les tests multi-environnements.[techrepublic2023virtualbox]



Figure 7. Logo VirtualBox

- **Grafana** : Grafana est un logiciel de surveillance et d'analyse de données open source. Il permet de créer des tableaux de bord personnalisés pour visualiser des métriques en temps réel et générer des rapports sur les performances des systèmes et des applications.[walsh2022grafana]



Figure 8. Logo Grafana

### 2.3.4 Outils de Collaboration et de Planification

Pour améliorer notre flux de travail et faciliter la communication et la planification au sein de l'équipe, nous avons utilisé plusieurs outils de collaboration.

- **Lucidchart** : Lucidchart est un outil de création de diagrammes en ligne qui permet de créer des diagrammes UML, des organigrammes, des diagrammes de réseau et bien plus encore. Il offre une interface intuitive et des fonctionnalités collaboratives, ce qui facilite la visualisation des processus et la communication des idées.



Figure 9. Logo Lucidchart

- **Kanboard** : Kanboard est un logiciel open source de gestion de projet basé sur la méthode Kanban. Il permet de visualiser les tâches, de suivre leur progression et de gérer les priorités de manière efficace. L'interface est simple et intuitive, facilitant la collaboration au sein de l'équipe.



Figure 10. Logo Kanboard

- **Google Drive** : Google Drive est un service de stockage en ligne qui permet de sauvegarder et de partager des fichiers en toute sécurité. Il offre des fonctionnalités de collaboration en temps réel, ce qui facilite le travail en équipe sur des documents, des tableurs et des présentations.



Figure 11. Logo Google Drive

- **Discord** : Discord est une plateforme de communication vocale, vidéo et textuelle largement utilisée pour les échanges en temps réel. Elle permet la création de serveurs privés pour l'équipe, facilitant les discussions, le partage de fichiers et la planification des réunions.



Figure 12. Logo Discord

## 2.4 Choix des Outils de Test et de Monitoring

### 2.4.1 Évaluation des Outils de Test de Charge

L'évaluation des outils de test de charge est cruciale pour garantir que la plateforme puisse simuler des conditions d'utilisation réalistes et identifier les points faibles du système. Plusieurs outils ont été considérés pour leur capacité à exécuter des tests de performance et de charge efficaces.

**Apache JMeter** est un outil open source largement utilisé pour les tests de charge. Sa flexibilité et son adaptabilité le rendent approprié pour une variété de scénarios de test. JMeter est prisé pour sa communauté active qui contribue avec de nombreuses extensions et plugins, augmentant ainsi ses capacités.

**Gatling**, un autre outil open source, se distingue par ses performances élevées et sa capacité à gérer des charges importantes. Sa conception basée sur le langage de programmation Scala permet une écriture de tests performante et une gestion efficace des ressources, en faisant une option solide pour les tests intensifs.

**LoadRunner** est une solution commerciale reconnue pour ses capacités d'analyse avancées. Bien que plus coûteux, il offre une suite intégrée de fonctionnalités qui peuvent être particulièrement utiles pour les entreprises à la recherche d'une solution complète et robuste. Cependant, il a été éliminé de la comparaison en raison de son caractère payant.

Pour mieux comprendre les différences entre ces outils, un tableau comparatif est présenté ci-dessous :

Critères		
Type	Open Source	Open Source
Date de Création	1998	2011
Langage de Scripting	Java	Scala
Interface Graphique	Oui	Non
Ligne de Commande	Oui	Oui
Protocoles Supportés	HTTP, FTP, JDBC, JMS, LDAP, etc.	HTTP, WebSocket
Courbe d'apprentissage	Peut être plus long pour les débutants en raison de l'interface graphique	Plus rapide pour les utilisateurs familiers avec Scala et les concepts de programmation fonctionnelle
Performances	Moins efficace pour les scénarios très intensifs en termes de charge, en particulier lorsqu'il est exécuté en mode GUI	Excellent performances grâce à son modèle asynchrone et au langage Scala
Documentation	Documentation complète et variée, y compris des tutoriels en ligne	Documentation en croissance, mais moins étendue que JMeter
Extensions et Plugins	Nombreux	Modérés
Communauté et Support	Large communauté	Communauté active
Coût	Gratuit	Gratuit

**Table 1.** Comparaison des Outils de Test de Charge [testsigma]

## 2.4.2 Évaluation des Outils de Monitoring

La visualisation des résultats de simulation est l'objectif principal de notre projet. Il est crucial de choisir une stratégie qui offre une interface conviviale et facile à comprendre pour garantir une analyse efficace des performances et des anomalies. Pour cela, nous avons évalué plusieurs approches de monitoring.

### ***Création d'un Dashboard from Scratch***

La première approche envisagée consistait à créer notre propre solution d'affichage des résultats en extrayant les données de rapport de JMeter et en les utilisant dans un dashboard développé avec Angular. Cette méthode offrirait une personnalisation totale, permettant d'adapter précisément l'interface et les fonctionnalités à nos besoins spécifiques.

### ***Utilisation de Solutions Prêts à l'emploi***

La seconde approche évaluée fut l'utilisation d'outils prêts à l'emploi comme Grafana. Grafana est un outil open source reconnu pour ses puissantes capacités de visualisation

et son intégration avec une multitude de sources de données. Il permet de configurer rapidement des tableaux de bord interactifs et personnalisés pour le suivi des performances en temps réel. Nous avons également considéré d'autres outils comme Kibana et Prometheus pour leurs capacités spécifiques.

Pour mieux comprendre les différences entre ces approches, un tableau comparatif est présenté ci-dessous :

Critères	Solution Personnalisée (Angular + JMeter)	Solution Prête à l'Emploi (Grafana, etc.)
<b>Flexibilité</b>	Très flexible, permet une personnalisation totale selon nos besoins spécifiques	Flexible avec de nombreuses options de personnalisation via des plugins et des intégrations
<b>Effort de Développement</b>	Élevé, nécessite du temps et des compétences en développement pour implémenter et maintenir la solution	Faible à modéré, configuration rapide avec une communauté active pour le support
<b>Intégration des Données</b>	Doit être développée manuellement pour chaque nouvelle source de données	Intègre facilement de multiples sources de données (bases de données, services cloud, etc.)
<b>Visualisation</b>	Dépend de l'effort de développement, possible de créer des visualisations sur mesure	Puissantes capacités de visualisation avec des tableaux de bord interactifs et dynamiques
<b>Communauté et Support</b>	Limitée à notre équipe de développement	Large communauté avec beaucoup de ressources, tutoriels et plugins
<b>Coût</b>	Temps et ressources de développement internes	Gratuit pour la version open source, coûts supplémentaires pour les fonctionnalités avancées (Grafana Enterprise)

**Table 2.** Comparaison des Solutions de Visualisation

### 2.4.3 Justification des choix effectués

#### Test de Charge

Après une évaluation approfondie des outils de test de charge, Apache JMeter a été sélectionné en raison de sa convivialité et de sa facilité d'utilisation, particulièrement pour les débutants.

**Convivialité et Accessibilité :** JMeter se distingue par son interface graphique intuitive, facilitant la configuration et l'exécution des tests. Cette interface est particulièrement avantageuse pour les utilisateurs qui ne sont pas familiers avec la programmation ou les concepts de tests de charge avancés. En offrant une approche visuelle, JMeter permet aux équipes de se concentrer sur la création de scénarios de test sans nécessiter une expertise technique approfondie.

**Documentation et Support Communautaire :** La vaste documentation de JMeter, combinée à une large communauté active, fournit des ressources précieuses pour les débutants.

Les utilisateurs peuvent facilement trouver des tutoriels, des guides et des réponses à leurs questions, ce qui accélère l'apprentissage et la résolution de problèmes.

**Polyvalence et Adaptabilité :** JMeter supporte une grande variété de protocoles et dispose de nombreuses extensions et plugins. Cette flexibilité permet de répondre à divers besoins de test de charge, rendant l'outil adaptable à de nombreux scénarios sans complexité additionnelle.

En résumé, bien que JMeter puisse être moins performant que d'autres outils pour des tests de charge très intensifs, il a été choisi principalement pour sa convivialité et sa facilité d'utilisation, qui sont des avantages cruciaux pour les débutants et les équipes de test recherchant une solution accessible et efficace.

#### **Outils de monitoring**

Dans notre projet, le choix des outils de monitoring a été guidé par la nécessité d'avoir une solution robuste, flexible et adaptée à notre environnement technique.

Après une évaluation approfondie, nous avons suivi l'approche de la Solution Prête à l'Emploi pour le monitoring. Bien que la création d'un dashboard from scratch ait été envisagée initialement pour maîtriser et personnaliser entièrement l'interface, cette approche présente des inconvénients potentiels. Elle nécessite un effort de développement important, pouvant entraîner des retards et des bugs. De plus, le maintien d'une solution personnalisée peut être chronophage et nécessiter des ressources dédiées. C'est pourquoi nous avons préféré une approche prête à l'emploi, en utilisant des outils tels que Grafana.

Le choix de Grafana parmi d'autres outils de monitoring comme Kibana et Prometheus s'est imposé après une évaluation approfondie. Grafana a été retenu pour plusieurs raisons clés. Tout d'abord, ses puissantes capacités de visualisation permettent de créer des tableaux de bord interactifs et personnalisés, offrant ainsi une interface conviviale et facile à comprendre pour l'analyse des performances et des anomalies. Cette capacité de visualisation en temps réel est cruciale pour notre projet, car elle permet de surveiller efficacement les résultats des simulations et de réagir rapidement en cas de problèmes.

De plus, Grafana offre une intégration avec une multitude de sources de données, ce qui est essentiel pour notre projet qui nécessite la collecte et l'analyse de données provenant de diverses sources. L'outil supporte de nombreux types de bases de données et services cloud, facilitant ainsi l'agrégation et la visualisation des données sans nécessiter un effort de développement supplémentaire pour chaque nouvelle source. En comparaison, bien que Kibana et Prometheus soient également des outils robustes, Grafana s'est distingué par sa flexibilité et sa convivialité, répondant parfaitement à nos besoins spécifiques.

## **2.5 Études Choix des Différents Types de Test**

Dans le cadre de l'optimisation de notre site e-commerce, il est essentiel d'étudier les différents types de tests de performance disponibles afin d'identifier les plus courants et pertinents. Cette section a pour objectif de fournir une analyse détaillée de ces tests, en mettant en lumière leur utilité et leur impact potentiel sur l'expérience utilisateur. Le tableau suivant résume les principaux types de tests de performance, en soulignant les caractéristiques, les avantages de chacun, ainsi que les métriques mesurées et les objectifs du test, afin de guider nos choix.

Type de Test	Caractéristiques	Avantages	Métriques Mesurées	Objectifs
<b>Test de Charge</b>	Simule un nombre spécifique d'utilisateurs simultanés sur le site	Évalue la capacité du site à gérer un nombre prévu d'utilisateurs	Temps de réponse, taux de requêtes	Assurer que le site peut gérer la charge attendue
<b>Test de Montée en Charge</b>	Augmente progressivement le nombre d'utilisateurs jusqu'à atteindre une limite	Identifie les points de dégradation de performance à différents niveaux de charge	Utilisation des ressources, seuils de dégradation	Déterminer la capacité maximale stable du site
<b>Test de Stress</b>	Soumet le site à une charge au-delà de ses capacités normales	Évalue la robustesse et la résilience du site sous des conditions extrêmes	Temps de réponse, taux d'erreurs	Identifier les points de rupture et les comportements sous charge extrême

**Table 3.** Résumé des principaux types de tests de performance.

## 2.6 Modélisation des scénarios de test

### 2.6.1 Définition des scénarios de test

La modélisation des scénarios de test est une étape cruciale dans l'évaluation des performances de notre site e-commerce. Il est essentiel de concevoir une variété de scénarios de test, allant des plus simples aux plus complexes, afin de simuler de manière réaliste la navigation des utilisateurs sur notre plateforme.

Les scénarios simples incluront des actions courantes telles que la navigation normale sur le site, la sélection de produits, et l'exploration des catégories. Ces scénarios permettent de tester la fluidité et la réactivité du site lors d'interactions de base.

Les scénarios plus complexes impliqueront des actions nécessitant une communication accrue avec la base de données, telles que la connexion à un compte utilisateur, la vérification du contenu du panier, et l'ajout de produits au panier. Ces scénarios permettent d'évaluer la performance du site sous des conditions d'utilisation plus intensives et réalistes.

Identifier et modéliser correctement ces scénarios est d'une importance capitale. Cela permet de couvrir un large éventail de cas d'utilisation, assurant ainsi que notre site peut offrir une expérience utilisateur fluide et efficace, même sous des charges variées et des conditions d'utilisation diverses. Une modélisation rigoureuse des scénarios de test aide également à anticiper les points de défaillance potentiels et à optimiser les performances du site pour répondre aux attentes des utilisateurs.

### 2.6.2 Identification des indicateurs de performance

Dans le cadre des tests de charge, nous avons défini et suivi des indicateurs clés de performance (KPI) pertinents. Ces KPI nous permettent de mesurer l'efficacité et la performance du site sous différentes charges.



Figure 13. Kpi

Parmi les KPI les plus couramment utilisés dans les tests de charge, on peut citer :

- **Temps de réponse** : Mesure le temps nécessaire pour qu'une requête soit traitée par le serveur et qu'une réponse soit envoyée à l'utilisateur.
- **Taux de requêtes par seconde (RPS)** : Indique le nombre de requêtes que le serveur peut traiter par seconde.
- **Utilisation des ressources** : Évalue l'utilisation du CPU, de la mémoire et des autres ressources du serveur sous charge.
- **Taux d'erreurs** : Mesure le pourcentage de requêtes qui échouent sous charge.

En suivant ces KPI, nous pouvons identifier les points faibles du site et prendre des mesures correctives pour améliorer ses performances. Les tests de charge, combinés à une analyse approfondie des KPI, jouent un rôle crucial dans l'assurance de la qualité et de la performance continue de notre site e-commerce.

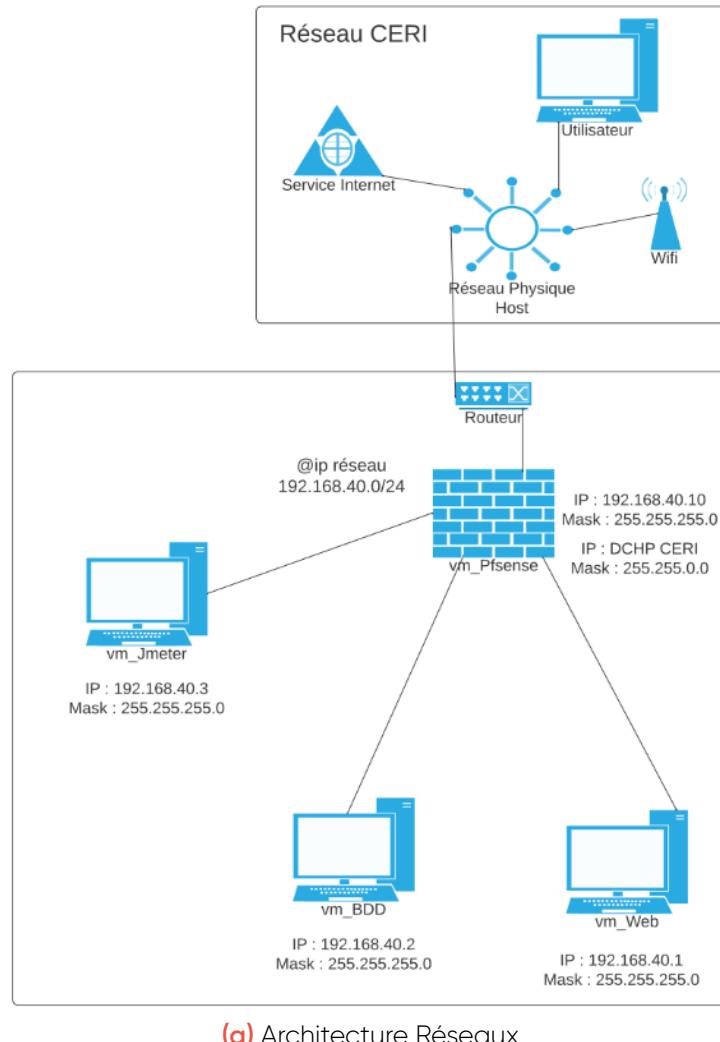
### 3 Phase de Réalisation (2ème Semestre)

#### 3.1 Développement

La phase de développement du projet consiste en la mise en place de l'infrastructure nécessaire et le développement du site e-commerce. Cette phase est cruciale pour établir un environnement de test réaliste qui réplique fidèlement l'architecture N-tiers utilisée par les systèmes e-commerce modernes.

##### 3.1.1 Mise en place de l'infrastructure avec les VMs

La première étape du développement implique la configuration des Machines Virtuelles (VMs). Chaque couche de l'architecture N-tiers est répliquée en utilisant des VMs distinctes pour les serveurs web, les serveurs d'applications et les bases de données. Cette configuration permet de simuler un environnement de production et de tester les interactions entre les différentes couches de manière réaliste.



(a) Architecture Réseaux

Les VMs sont créées et configurées avec les systèmes d'exploitation et les logiciels requis. Par exemple, les commandes d'installation et de configuration sont exécutées pour mettre en place un serveur web, des environnements d'exécution, et la base de données MySQL.

Pour les besoins de test, une VM dédiée à JMeter est également configurée. Cette VM permet de créer et d'exécuter des scénarios de test simulant des utilisateurs réels interagissant avec le système, ce qui aide à évaluer les performances et la stabilité du site e-commerce sous différentes charges.

La configuration du réseau interne est essentielle pour assurer une communication sécurisée entre les différentes couches de l'architecture. Les paramètres de réseau sont ajustés pour permettre un trafic fluide et sécurisé entre le serveur web, le serveur d'application et la base de données. Les VMs sont configurées pour communiquer efficacement entre elles, garantissant ainsi que les tests de performance reflètent fidèlement les conditions réelles d'utilisation du système.



(a) Configuration Réseaux

### 3.1.2 Conception et Développement du site e-commerce

#### Conception :

Nous allons inclure un diagramme de classe pour illustrer de manière claire et structurée la conception de l'architecture de notre application e-commerce. En détaillant les entités principales, leurs attributs et les relations entre elles, il offre une vue d'ensemble compréhensible du système, facilitant ainsi la communication entre les différentes parties prenantes.

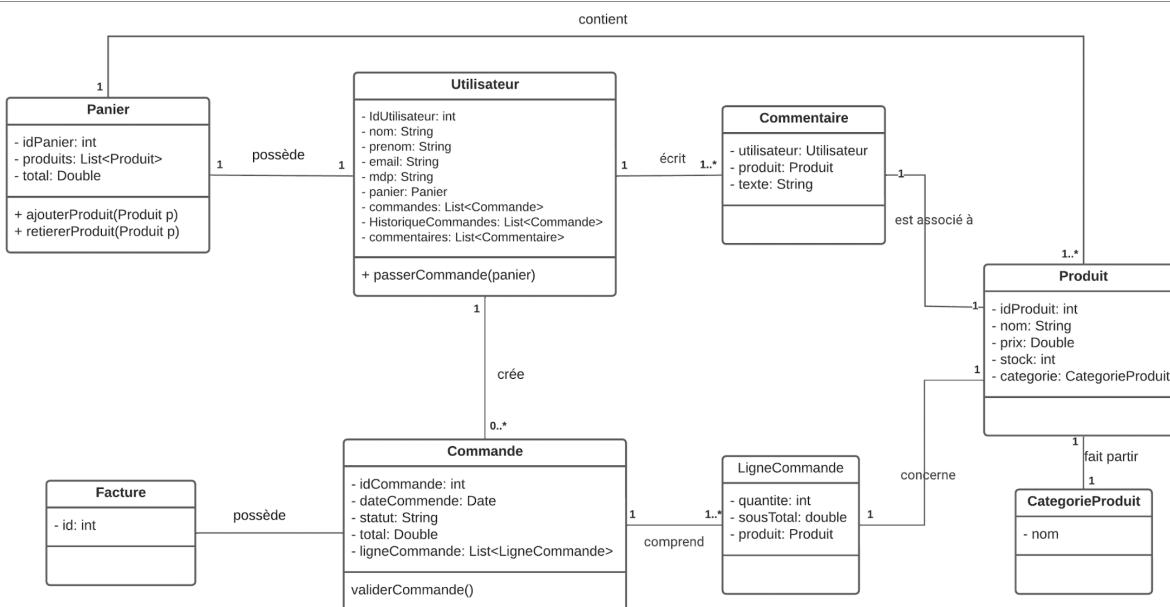


Figure 16. diagramme de classe

Voici une description concise de chaque classe avec leurs relations avec les autres classes :

- **User (Utilisateur)**

- **Champs** : `userId, username, password, email, role, active, cart, orders`

- **Relations :**
  - A un panier (**Cart**) en relation **1 à 1**.
  - A plusieurs commandes (**Order**) en relation **1 à plusieurs**.
- **Cart (Panier)**
  - **Champs :** id, user, cartItems
  - **Relations :**
    - Appartient à un utilisateur (**User**) en relation **1 à 1**.
    - Contient plusieurs articles du panier (**CartItem**) en relation **1 à plusieurs**.
- **CartItem (Article du Panier)**
  - **Champs :** id, quantity, product, cart
  - **Relations :**
    - Appartient à un panier (**Cart**) en relation **plusieurs à 1**.
    - Correspond à un produit (**Product**) en relation **1 à 1**.
- **Order (Commande)**
  - **Champs :** id, user, status, orderDetails, paymentMethod, orderDate, shippingAddress, orderTotal
  - **Relations :**
    - Appartient à un utilisateur (**User**) en relation **1 à 1**.
    - Contient plusieurs détails de commande (**OrderDetails**) en relation **1 à plusieurs**.
- **OrderDetails (Détails de la Commande)**
  - **Champs :** id, order, product, unitPrice, quantity, totalAmount
  - **Relations :**
    - Appartient à une commande (**Order**) en relation **plusieurs à 1**.
    - Correspond à un produit (**Product**) en relation **1 à 1**.
- **Product (Produit)**
  - **Champs :** productId, name, description, quantity, price, categories, images
  - **Relations :**
    - Peut appartenir à plusieurs catégories (**Category**) en relation **plusieurs à plusieurs**.
    - Peut avoir plusieurs images (**ProductImage**) en relation **1 à plusieurs**.
    - Apparaît dans plusieurs articles du panier (**CartItem**) en relation **1 à plusieurs**.
    - Apparaît dans plusieurs détails de commande (**OrderDetails**) en relation **1 à plusieurs**.
- **Category (Catégorie)**
  - **Champs :** categoryId, name, description, parentCategory, childCategories, products
  - **Relations :**
    - Peut contenir plusieurs produits (**Product**) en relation **plusieurs à plusieurs**.
    - Peut avoir plusieurs sous-catégories (**Category**) et peut être une sous-catégorie d'une autre catégorie, en relation **1 à plusieurs** avec elle-même.
- **ProductImage (Image du Produit)**
  - **Champs :** id, imageUrl, product
  - **Relations :**
    - Appartient à un produit (**Product**) en relation **plusieurs à 1**.
- **Facture (Invoice)**
  - **Champs :** id, order, invoiceDate, paymentMethod, paymentStatus, totalAmount
  - **Relations :**
    - Appartient à une commande (**Order**) en relation **1 à 1**.

**Développement :** La coordination entre le développement frontend et backend est cruciale pour assurer une intégration harmonieuse des fonctionnalités et offrir une expérience utilisateur fluide et cohérente.

#### Front-end :

Le développement du frontend a été réalisé en utilisant des technologies modernes telles qu'Angular, permettant la création d'une interface utilisateur réactive et dynamique.

L'interface utilisateur est divisée en composants réutilisables, facilitant ainsi la maintenance et l'extensibilité de l'application. Chaque composant est responsable d'une partie spécifique de l'interface, comme l'affichage des produits, la gestion du panier, ou la réalisation des commandes.

Pour assurer une interaction fluide avec l'API RESTful du backend, des services Angular ont été utilisés pour les appels HTTP. Les données récupérées du serveur sont gérées de manière efficace et sont affichées en temps réel grâce au state management d'Angular.

En outre, une attention particulière a été portée à la conception responsive, garantissant que le site est pleinement fonctionnel et accessible sur divers appareils, tels que les ordinateurs de bureau, les tablettes, et les smartphones.

Samsung Galaxy S8+ ▾ 360 x 740 75% ▾ No throttling ▾

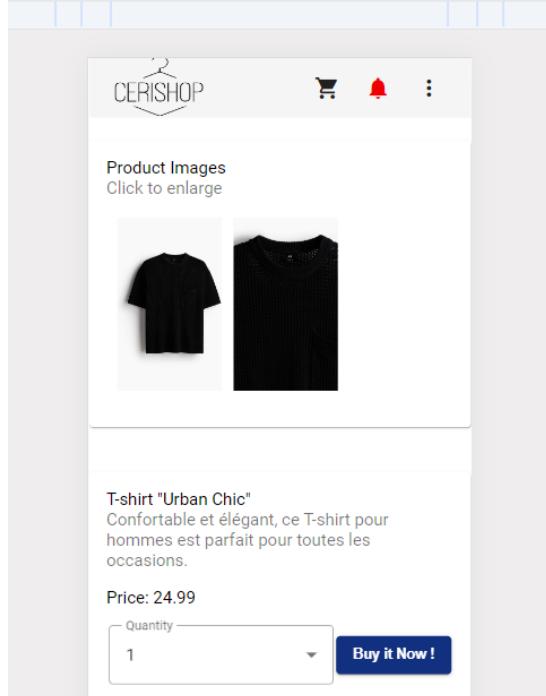


Figure 17. le site web sur un smartphone

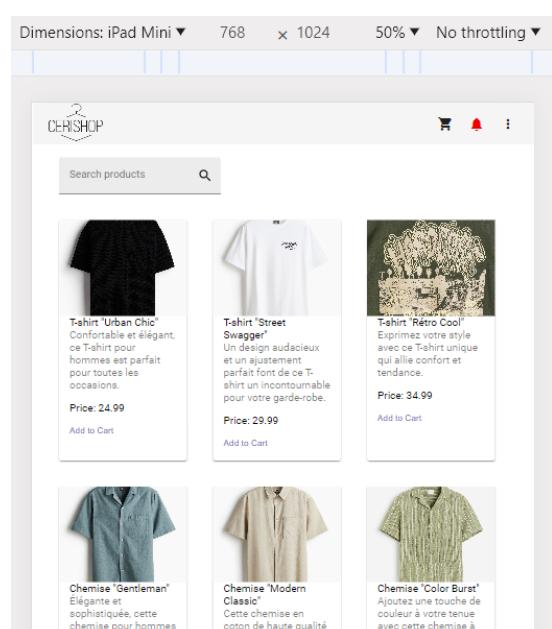


Figure 18. le site web sur une tablette

**Fonctionnalités** Le frontend du site e-commerce offre plusieurs fonctionnalités clés, telles que :

- **Affichage des produits :** Les utilisateurs peuvent parcourir et consulter des produits avec des informations détaillées comme le nom, la description, le prix, et des images.
- **Recherche et filtrage :** Une barre de recherche avancée et des filtres permettent aux utilisateurs de trouver rapidement les produits qu'ils recherchent.

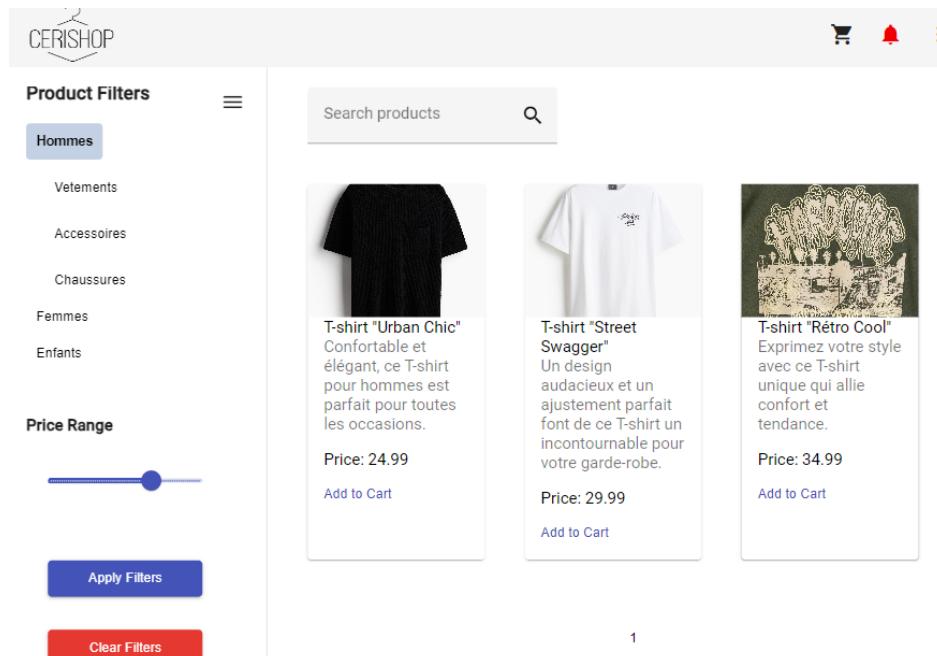


Figure 19. Recherche et filtrage des produits

**Gestion du panier :** Chaque utilisateur, qu'il soit connecté ou non, dispose d'un panier où il peut ajouter des produits, les modifier. Les utilisateurs connectés bénéficient de la fonctionnalité supplémentaire de sauvegarde de l'état de leur panier entre les sessions, ce qui leur permet de retrouver les articles précédemment ajoutés lors de leur prochaine visite et de finaliser leur commande en toute simplicité.

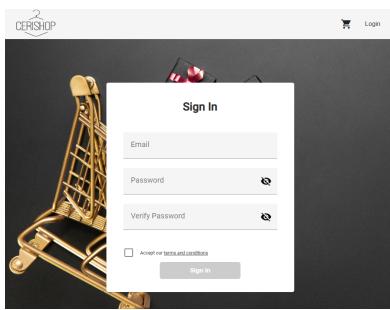
The screenshot shows the Cerishop shopping cart page. At the top, there is a header with the Cerishop logo and a navigation bar with icons for a cart, notifications, and more. The main title is "Shopping Cart". Below the title is a table showing the items in the cart:

Product Name	Product Image	Product Price	Quantity	Actions
T-shirt "Rétro Cool"		34.99	1	
Robe "Floral Fantasy"		84.99	1	
Sweatshirt "Casual Cool"		59.99	1	

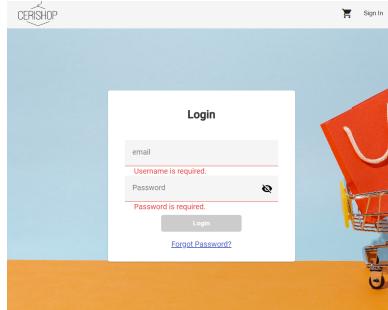
Below the table, there is a checkbox for "Add Delivery Fee (5€)" and a total amount of "Total: 179.97 €". At the bottom is a green "Checkout" button.

Figure 20. Panier

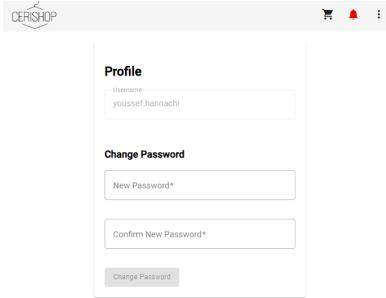
- Profil utilisateur :** Les utilisateurs peuvent s'enregistrer, se connecter, gérer leurs informations personnelles.



**Figure 21.** interface Sign-in



**Figure 22.** interface Login



**Figure 23.** interface profile

#### Back-end :

Le serveur back-end a été développé en utilisant Spring Boot et répond parfaitement à tout les besoins identifiés dans la phase d'analyse.

L'architecture RESTful de l'API a été soigneusement élaborée pour offrir une interaction intuitive et structurée avec le client. Chaque ressource de l'application est accessible via des endpoints HTTP clairs et définis, permettant des opérations telles que la récupération des produits, la gestion des utilisateurs...

Pour assurer la sécurité de l'application, une couche de sécurité a été mise en place en utilisant Spring Security, un module de sécurité complet et extensible. L'authentification et l'autorisation des utilisateurs sont gérées à l'aide de JSON Web Tokens (JWT), ce qui permet une validation sécurisée et stateless des utilisateurs. Les tokens JWT sont générés lors de la connexion et doivent être inclus dans les en-têtes des requêtes HTTP pour accéder aux ressources protégées. Cela garantit que seules les demandes authentifiées et autorisées peuvent accéder aux fonctionnalités critiques de l'application.

#### 3.1.3 Crédit et intégration de la base de données

Spring Boot facilite considérablement le développement d'applications grâce à son intégration avec Spring Data JPA, permettant de définir des classes d'entité directement en Java. En annotant ces classes avec des annotations telles que `@Entity`, `@Table`, `@Id`, et `@GeneratedValue`, nous pouvons spécifier les correspondances entre les classes Java et les tables de la base de données. Une fois les classes définies, Spring Boot utilise ces annotations pour générer automatiquement les schémas de base de données correspondants lors de l'initialisation de l'application. Cette fonctionnalité élimine le besoin d'écrire manuellement des scripts SQL pour la création de tables, simplifiant ainsi le processus de développement.

Après avoir configuré les entités et lancé l'application, nous avons pu peupler notre base de données avec des données réelles. Voici notre base de données mySql .

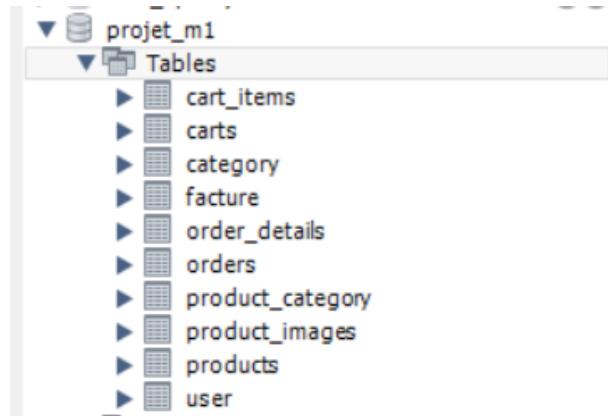


Figure 24. Tables de la base MySql

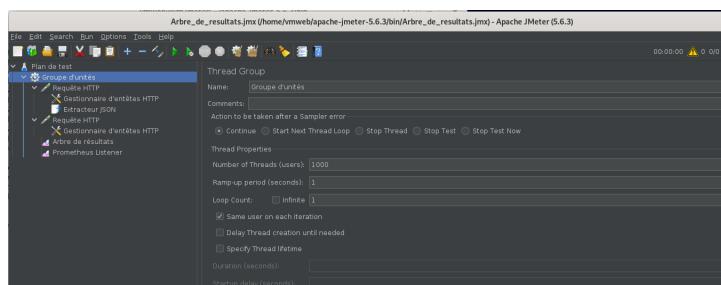
## 3.2 Mise en Œuvre des Outils de Test et de Monitoring

La mise en œuvre des outils de test et de monitoring est essentielle pour évaluer précisément la performance du système. Cette section détaille les étapes de configuration de JMeter pour les tests de charge ainsi que l'intégration de Grafana et Prometheus pour le monitoring des performances.

### 3.2.1 Configuration de JMeter pour les tests de charge

La configuration de JMeter pour les tests de charge commence par l'installation de JMeter sur des machines virtuelles (VMs) dédiées aux tests. Une fois installé, JMeter est configuré pour exécuter des scénarios de test basés sur les cas d'utilisation identifiés, tels que la navigation, les transactions et la gestion des utilisateurs.

Les paramètres de test, y compris le nombre d'utilisateurs virtuels, la période de montée en charge (ramp-up period) et la durée des tests, sont définis pour simuler des conditions d'utilisation réalistes. Cette configuration permet de créer des environnements de test robustes, capables de reproduire des charges de travail similaires à celles rencontrées en production.



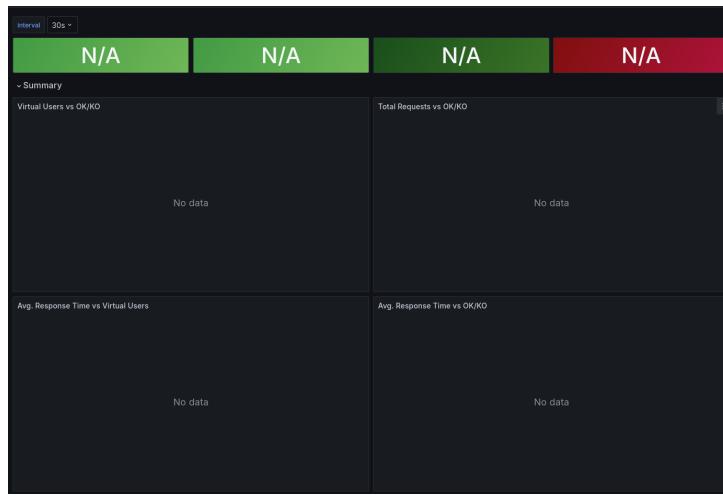
(a) Interface JMeter

### 3.2.2 Intégration de Grafana pour le monitoring des performances

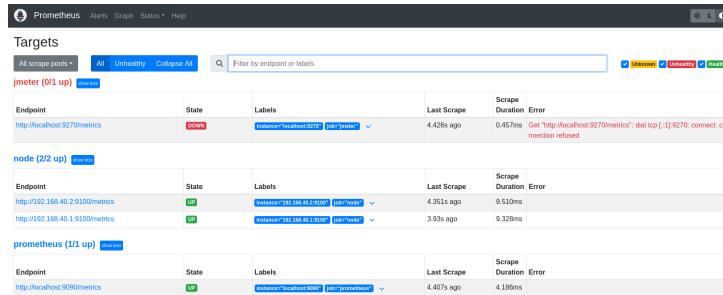
L'intégration de Grafana pour le monitoring des performances commence par l'installation et la configuration de Grafana sur une VM dédiée. Grafana est un outil puissant pour la visualisation des données de performance en temps réel, et son intégration avec Prometheus est particulièrement bénéfique pour le monitoring à grande échelle.

Prometheus est configuré pour collecter et stocker des métriques de performance provenant de diverses sources. Pour surveiller les performances du système, nous utilisons un outil appelé node\_exporter, qui est un exportateur Prometheus. Node\_exporter est installé sur chaque serveur (ou VM) à surveiller. Il collecte des métriques système telles que l'utilisation du CPU, de la mémoire, du disque et du réseau, et les expose sous une forme que Prometheus peut scraper.

Prometheus est installé sur une VM dédiée et configuré pour scraper les métriques fournies par node\_exporter. Le fichier de configuration de Prometheus (prometheus.yml) inclut les endpoints des node\_exporters déployés sur les serveurs cibles, permettant ainsi à Prometheus de récupérer régulièrement les données de performance des serveurs spécifiés.



(a) Grafana



(b) Prometheus

Grafana est ensuite configuré pour se connecter à Prometheus comme source de données. Cela se fait dans l'interface de configuration de Grafana, où Prometheus est ajouté en tant que source de données en fournissant l'URL du serveur Prometheus. Une fois la connexion établie, des tableaux de bord sont créés dans Grafana pour visualiser les données collectées. Ces tableaux de bord peuvent inclure des graphiques pour l'utilisation du CPU, de la mémoire, les I/O, etc. Grafana permet également de configurer des alertes basées sur les métriques de Prometheus, déclenchant des notifications en cas d'anomalies.

En résumé, l'intégration de Grafana et Prometheus, avec l'aide de node\_exporter, permet de créer une solution de monitoring puissante et extensible. Cette configuration assure une surveillance détaillée et en temps réel des performances du système, facilitant l'identification rapide des problèmes et l'optimisation continue des performances.

### 3.3 Exécution des Tests de Charge

#### 3.3.1 Déroulement des tests de charge

Dans cette section nous allons parler plus des test performé avec Jmeter en illustrant des captures d'écran de ce qu'on simule du coté client à savoir notre page web du site.

##### Scénario 1 :

C'est le scénario le plus simple , on simule des utilisateurs qui se rendent sur notre site, cette action déjà résulte en une communication entre le front et le back ainsi que la base de données puisque la page d'accueil permet d'afficher tout les produit ainsi que les catégories présents sur la base de données. Dans ce test on va simuler une charge élevée d'utilisateur pour visualiser comment notre site réagit.



Figure 27. Page d'accueil testée

##### Scénario 2 :

Ce deuxième scénario, plus complexe, vise à simuler la navigation normale d'un utilisateur lambda sur un site. Celui-ci navigue, regarde les produits, essaie les filtres, choisit des catégories, puis ferme finalement le site. Ce test permet de simuler un scénario réel qui peut représenter les actions d'un utilisateur typique sur un site e-commerce. Dans Jmeter on utilise plus de fonctionnalités aussi puisque dans ce test on extrait les réponses dans des variables pour les utiliser plus tard, des fichiers csv aussi pour que tout les threads n'utilisent pas les même données.

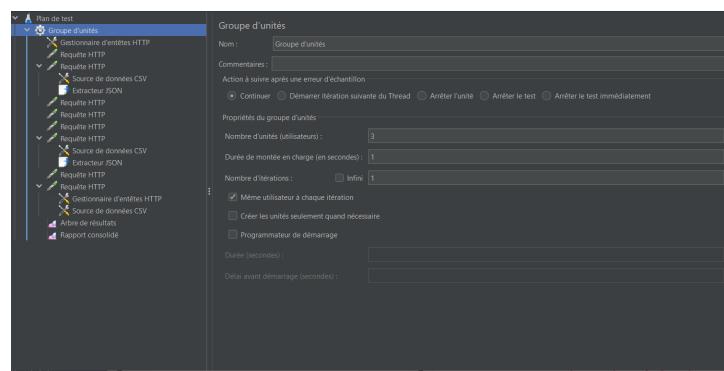
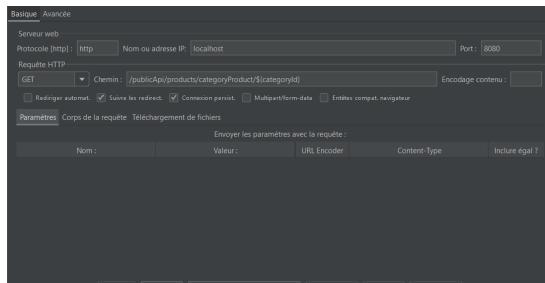
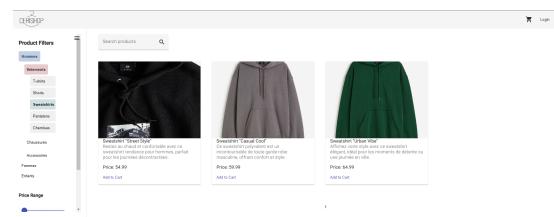


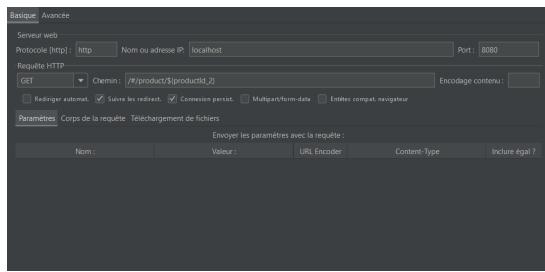
Figure 28. Configuration Jmeter du scénario de test 2



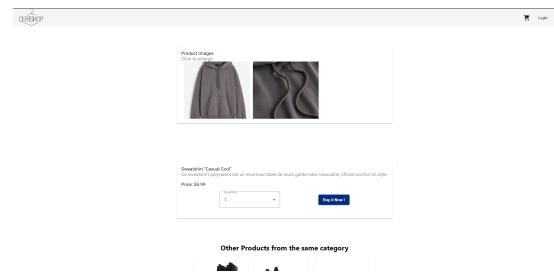
(a) Configuration category du scénario 2



(b) Page category dans le site testée



(a) Configuration product du scénario 2



(b) Page product dans le site testée

**Scénario 3 :** Dans ce troisième scénario, ce que l'on essaie de faire, c'est de simuler un utilisateur qui vient sur le site, se connecte, ajoute des produits à son panier après avoir navigué, vérifie les produits présents dans son panier, supprime un ou plusieurs produits et se déconnecte. Ce scénario complexe permet de tester plusieurs fonctionnalités de notre site. Il communique avec la base de données pour la connexion et passe par la couche de sécurité. Une fois connecté, les appels aux API du site sont sécurisés ; il faudra donc utiliser le token JWT fourni après une connexion réussie à l'utilisateur. Dans ce test, nous utilisons également des extracteurs dans JMeter pour récupérer les JWT, ainsi que des fichiers CSV pour nous connecter à chaque fois avec un compte différent pour chaque thread.

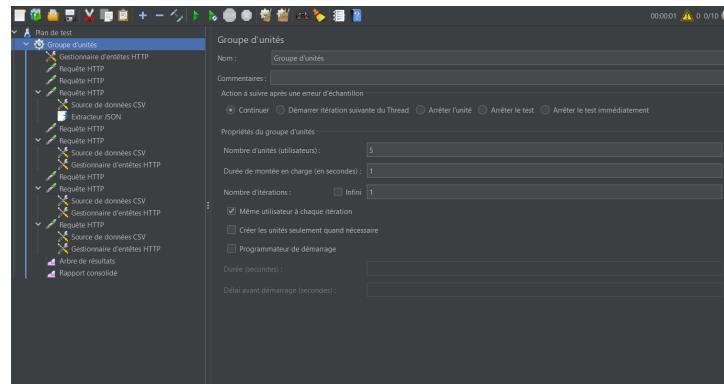
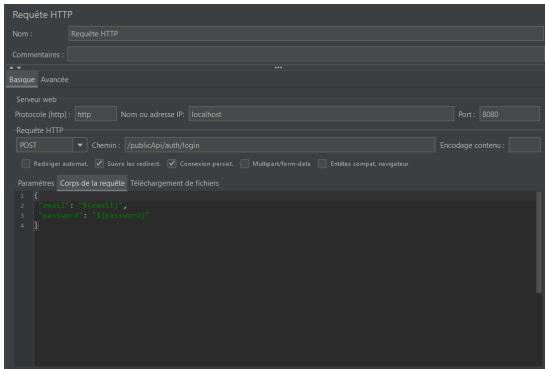
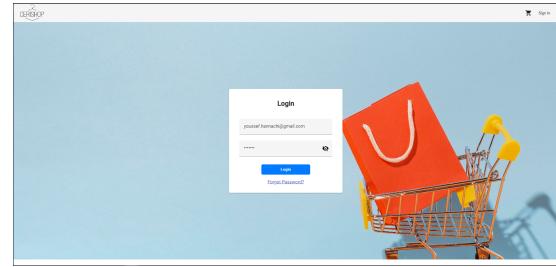


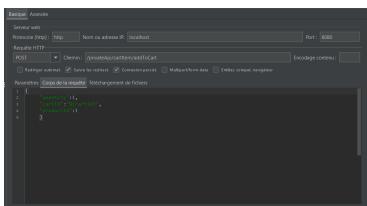
Figure 31. Configuration Jmeter scénario 3



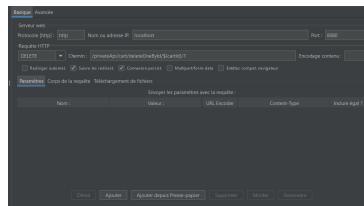
(a) Configuration login du scénario de test3



(b) Page login dans le site testé



(a) Configuration panier du scénario de test 3



(b) Configuration suppression élément du panier scénario test 3



(c) Page panier sur le site testé

### Note

Voici comment la configuration du token dans l'en-tête des requêtes sécurisées est réalisée.

Entêtes stockées	
	Nom :
Authorization	Bearer \${token}

Figure 34. Configuration en-tête avec JWT

### 3.3.2 Collecte et analyse des résultats

Pendant ces tests, JMeter est utilisé en concordance avec Prometheus pour capturer des données de performance telles que le temps de réponse, le taux de réussite des transactions et le débit. Grafana, intégré pour le monitoring en temps réel, surveille l'utilisation des ressources, notamment le CPU, la mémoire et les I/O du serveur. Ces outils collectent des données précieuses qui sont ensuite analysées pour identifier les tendances, les anomalies et les goulots d'étranglement.

### 3.3.3 Ajustements et optimisation

L'analyse des résultats permet de comparer les performances observées avec les objectifs définis. Si des points faibles sont identifiés, des ajustements sont effectués, comme l'optimisation des requêtes SQL, l'amélioration du code ou l'ajustement des configurations serveurs. Chaque modification est suivie d'une réexécution des tests pour vérifier l'efficacité des optimisations apportées. Cette approche itérative assure que toutes les optimisations conduisent à une amélioration tangible des performances du système.

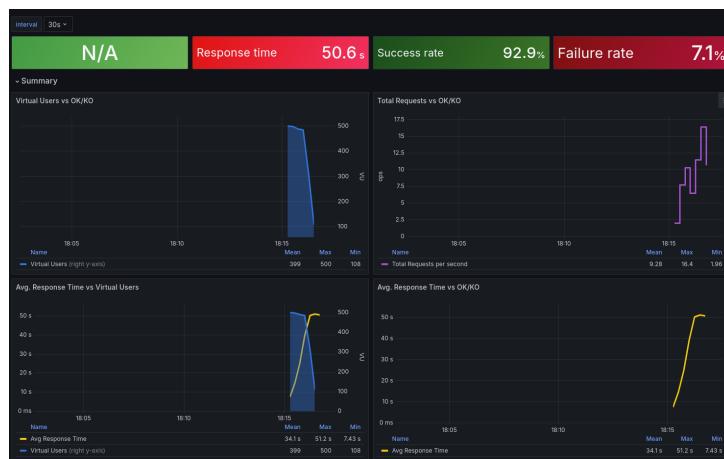
En conclusion, l'exécution rigoureuse des tests de charge selon des scénarios réalistes et variés permet d'évaluer de manière exhaustive les capacités du système. Cette phase

critique assure que le site e-commerce peut gérer des charges élevées tout en maintenant des performances optimales pour les utilisateurs, qu'ils soient anonymes ou authentifiés.

### 3.4 Analyse des Résultats

#### 3.4.1 Présentation des résultats des tests

Les résultats des tests de charge sont présentés de manière détaillée sous forme de rapports comprenant des graphiques et des tableaux comparatifs. Ces rapports fournissent une vue d'ensemble des performances du système en mettant en évidence à la fois les points forts et les points faibles. Les graphiques illustrent les tendances observées pendant les tests, tandis que les tableaux comparatifs permettent de comparer les performances sous différentes conditions de charge. Cette présentation visuelle aide à comprendre comment le système réagit à des charges variées et à identifier les domaines nécessitant des améliorations.

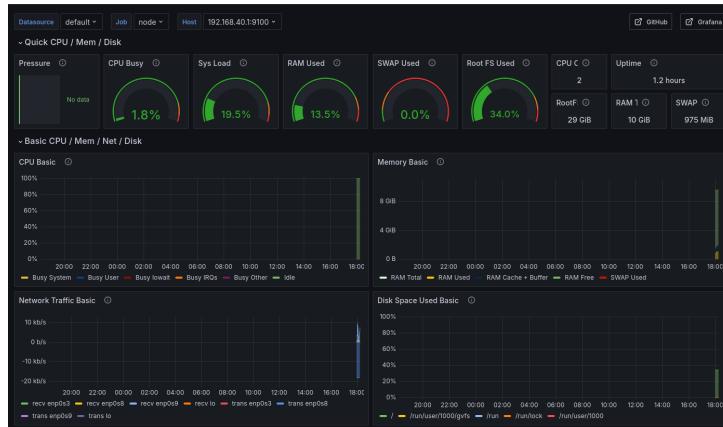


(a) Graphana avec résultats des tests

#### 3.4.2 Analyse des performances

L'évaluation des performances globales du système se concentre sur plusieurs aspects clés. Les temps de réponse sont mesurés pour évaluer la rapidité avec laquelle le système répond aux requêtes des utilisateurs. La capacité de traitement est analysée pour déterminer combien de transactions le système peut gérer simultanément sans dégradation des performances. La stabilité du système est également évaluée en observant son comportement sous des charges constantes et variables.

En outre, les taux d'erreur sont analysés pour identifier les causes sous-jacentes des échecs et des anomalies. Cette analyse détaillée permet de comprendre les limites actuelles du système et d'établir des benchmarks pour les futures optimisations.



(a) Les différentes métriques

### 3.4.3 Identification des goulots d'étranglement et propositions d'améliorations

Le diagnostic des problèmes de performance se concentre sur l'identification des goulots d'étranglement au sein du système. Ces goulots peuvent se situer au niveau des serveurs, de la base de données, ou des applications elles-mêmes. Une fois identifiés, des propositions d'améliorations spécifiques sont formulées pour résoudre ces problèmes. Cela peut inclure l'optimisation des requêtes SQL pour améliorer l'efficacité de la base de données, l'amélioration du code pour augmenter la performance de l'application, ou l'augmentation des ressources matérielles pour gérer une charge plus élevée. Chaque proposition est testée pour vérifier son efficacité avant d'être mise en œuvre de manière permanente.

En résumé, l'analyse des résultats des tests est essentielle pour comprendre les performances actuelles du système, identifier les domaines nécessitant des améliorations, et proposer des solutions concrètes pour optimiser le fonctionnement global du site e-commerce. Cette phase assure que le système peut fonctionner de manière fiable et efficace sous des conditions de charge réalistes.

## 4 Difficultés Rencontrées

### 4.1 Méconnaissance des logiciels et leur fonctionnement

L'un des défis majeurs rencontrés au cours du projet a été la méconnaissance initiale des logiciels utilisés, notamment JMeter, Grafana, et Prometheus. Bien que ces outils soient puissants et largement adoptés dans l'industrie, leur utilisation efficace nécessite une compréhension approfondie de leurs fonctionnalités et de leur configuration.

Au début, l'équipe a dû investir un temps considérable pour se former à ces outils, comprendre leur architecture, et maîtriser leurs interfaces. Par exemple, la création de scénarios de test complexes avec JMeter a exigé une phase d'apprentissage pour manipuler les éléments de test et les paramètres appropriés. De même, la configuration de tableaux de bord personnalisés dans Grafana et la collecte de métriques avec Prometheus ont nécessité une exploration approfondie de la documentation et de nombreux essais et erreurs.

### 4.2 Lien entre Grafana et Prometheus

Un autre défi important a été l'intégration de Grafana avec Prometheus pour le monitoring en temps réel. Bien que ces deux outils soient conçus pour fonctionner ensemble, la mise en place du lien entre eux a présenté des obstacles techniques. La configuration de Prometheus pour collecter les métriques système à l'aide de node-exporter, puis leur exportation vers Grafana pour visualisation, a nécessité une compréhension détaillée des configurations de chaque outil. Les difficultés incluaient la mise en place correcte des endpoints de collecte de données et la gestion des permissions et des accès.

De plus, la création de tableaux de bord pertinents dans Grafana, adaptés aux besoins spécifiques du projet, a demandé une connaissance approfondie des requêtes PromQL (Prometheus Query Language).

### 4.3 Connexion entre les différentes VM

La connexion et la communication entre les différentes machines virtuelles (VMs) constituaient un autre défi de taille. Étant donné que l'architecture N-tiers repose sur une interaction fluide entre le serveur web, le serveur d'applications et la base de données, assurer une communication fiable et sécurisée entre ces composants était crucial. Des problèmes de connectivité, tels que des pings échoués et des configurations réseau incorrectes, ont initialement entravé le progrès.

La configuration du réseau interne pour permettre la communication sécurisée entre les différentes couches de l'architecture a nécessité des ajustements minutieux des paramètres réseau et des vérifications de sécurité. De plus, la gestion des pare-feu et des règles de sécurité pour autoriser les communications nécessaires tout en empêchant les accès non autorisés a ajouté une couche supplémentaire de complexité.

### 4.4 Problèmes pour l'app web entre le front et le back

Le développement et l'intégration de l'application web ont également rencontré des difficultés, notamment dans la communication entre le front-end et le back-end. Utilisant Angular pour le front-end et Spring Boot pour le back-end, des problèmes de compatibilité et de communication sont apparus. Les API RESTful mises en place pour permettre la communication entre les deux parties ont parfois rencontré des problèmes de

synchronisation et de gestion des sessions.

Par exemple, des erreurs de CORS (Cross-Origin Resource Sharing) ont empêché les requêtes front-end d'atteindre le back-end. De plus, des problèmes de gestion des sessions utilisateur et de maintien de la sécurité des transactions ont nécessité des révisions fréquentes du code et des ajustements des configurations. La résolution de ces problèmes a impliqué une collaboration étroite entre les développeurs front-end et back-end, ainsi que des tests rigoureux pour assurer que toutes les fonctionnalités de l'application fonctionnent de manière transparente.

En résumé, bien que le projet ait rencontré plusieurs défis techniques et logistiques, chacun de ces obstacles a offert des opportunités d'apprentissage et de renforcement des compétences de l'équipe. Les solutions mises en place pour surmonter ces difficultés ont non seulement permis de mener le projet à bien, mais ont également fourni une base solide pour les futures initiatives de développement et de performance.

## 5 Conclusion et Perspectives

### 5.1 Résumé des travaux réalisés

Le projet s'est déroulé en plusieurs phases, chacune visant à mettre en place une plateforme de simulation de tests pour évaluer la performance d'une architecture N-tiers de type e-commerce. Tout d'abord, une étude approfondie a été réalisée pour définir les besoins et évaluer les solutions de test de performance disponibles sur le marché. Ensuite, une maquette a été préparée pour créer un environnement de test réaliste comprenant une infrastructure réseau, système et applicative.

La phase de développement a impliqué la configuration des machines virtuelles, le développement du front-end avec Angular et du back-end avec Spring Boot, ainsi que l'intégration de la base de données MySQL. Les outils de test et de monitoring, tels que JMeter pour les tests de charge et Grafana et Prometheus pour le monitoring des performances, ont été configurés et intégrés. Les tests de charge ont été exécutés selon des scénarios prédéfinis, et les résultats ont été analysés pour identifier les points forts et les faiblesses du système.

Cette évaluation rigoureuse a permis d'atteindre les objectifs initiaux du projet, en fournant une compréhension claire des performances du système sous diverses conditions de charge.

### 5.2 Bilan par rapport aux objectifs initiaux

L'évaluation de la réussite du projet se fait par rapport aux objectifs initiaux fixés en termes de performance, scalabilité et sécurité. Les résultats obtenus servent à montrer que le système est capable de gérer ou non efficacement des charges importantes, avec des temps de réponse acceptables et une capacité de traitement adéquate. Dans notre cas, le système parvient à soutenir des charges allant d'une centaine d'utilisateurs tout en maintenant des performances et temps de réponses acceptables.

La scalabilité a été démontrée par la capacité du système à augmenter les ressources pour gérer des pics de trafic en minimisant les dégradations de performance. En termes de sécurité, les mesures mises en place ont assuré la protection des données utilisateur et la conformité aux normes applicables.

Toutefois, certains écarts par rapport aux attentes initiales ont été identifiés, notamment dans les domaines de l'optimisation des requêtes et de l'amélioration du code ainsi que la performance du système. Ces écarts fournissent des leçons précieuses pour les projets futurs et les efforts d'optimisation continue.

### 5.3 Perspectives d'amélioration et de prolongement du projet

Pour l'avenir, plusieurs pistes d'amélioration ont été identifiées afin d'optimiser davantage les performances du système. Une des premières étapes serait d'intégrer de nouvelles fonctionnalités pour améliorer l'efficacité des tests et le monitoring. Par exemple, l'ajout de tests de performance supplémentaires, tels que des tests de montée en charge prolongée, pourrait fournir des données plus détaillées sur la résilience du système.

L'adoption de nouvelles technologies de monitoring et de gestion des performances, comme des outils avancés de gestion de logs ou des systèmes de monitoring distribués, pourrait également offrir des avantages significatifs en termes de précision et de réactivité.

De plus, l'extension des tests à d'autres scénarios d'utilisation, comme les périodes de forte activité, permettrait de mieux comprendre le comportement du système dans des situations variées et de garantir sa robustesse.

La mise en place d'un processus de monitoring et d'optimisation continue des performances est essentielle pour garantir que le système reste performant et fiable à long terme. Cela inclut l'utilisation régulière des outils de monitoring, l'analyse des données collectées, et l'ajustement des configurations en fonction des besoins changeants. La création de tableaux de bord plus détaillés et personnalisés dans Grafana pourrait améliorer la visualisation et l'interprétation des données de performance en temps réel.

En conclusion, le projet a permis de développer une plateforme robuste et performante pour l'évaluation des systèmes e-commerce. Les perspectives d'amélioration sont prometteuses, avec des possibilités d'optimisation continue et l'intégration de nouvelles technologies pour maintenir et améliorer la performance du système.

## 6 Bibliographie

Pour installer Apache JMeter sur Linux, vous pouvez suivre ce guide détaillé disponible sur [GeeksforGeeks](#). Il fournit des instructions pas à pas pour la configuration de JMeter sur une machine Linux.

L'intégration de JMeter avec Prometheus et Grafana est expliquée dans cet article sur [GeeksforGeeks](#). Ce guide vous aide à configurer JMeter pour envoyer des métriques à Prometheus et visualiser les résultats dans Grafana.

Pour une introduction et des premiers pas avec Prometheus, vous pouvez consulter la documentation officielle sur le site de [Prometheus](#). Cette documentation offre un aperçu des concepts fondamentaux et des étapes pour démarrer avec Prometheus.

Pour une intégration entre Prometheus et Graphana, vous pouvez consulter la documentation officielle sur le site de [Grafana](#).