Ecole Hassania
des Travaux Publics

المدرسة الحسنية للأشغال العمومية

☑ PFE
☐ SI
☐ SIP

Rapport de Stage

Sujet:

Conception et réalisation d'un outil de Simulation de Service d'Auto Scaling Cloud pour la SG ATS

résumé :

Le projet consiste en la réalisation d'un simulateur de service de Scaling relative aux services qui sont exposés sur le cloud et utilisés par les différents collaborateurs de la Société Générale Africa Technogies & Services. L'outil permet de simuler le service de Scaling dans le but d'améliorer son efficacité et assurer la fiabilité et la haute performance des différents services utilisés sur le Cloud.

Réalisé par :

Zouaid Omar ***3GI***

Encadré par :

Pr. Malika Addou - EHTP

M. Tarik Douiyeh - SG ATS

Année universitaire : 2019- 2020

SOCIETE GENERALE
Corporate & Investment Banking
Africa Technologies & Services

Société Générale Africa technologies & Services :

Résidence Walili Street, 42 Boulevard Abdelmoumen, Casablanca 20250.

Remerciements

Avant d'entamer ce rapport, je tiens à témoigner ma profonde gratitude à toutes les personnes qui ont participé de près ou de loin à l'élaboration de ce travail.

Je souhaite faire part de ma reconnaissance en premier lieu à mon encadrante à l'EHTP le Pr. Malika ADDOU qui a fait preuve de disponibilité à chaque fois que j'avais besoin de son soutien. Son excellent suivi et ses remarques très enrichissantes m'ont été d'un appui considérable.

Mes sincères remerciements s'adressent aussi à mon encadrant externe M. Tarik DOUIYEH, pour son dévouement et ses conseils lucides et pertinents qu'il n'a cessé de me prodiguer tout au long de ce projet.

J'exprime mes vifs remerciements et respects à tous les membres de l'équipe RDWS, spécialement M. Youssef LGhOUMAIGUI, votre assistance a été d'une très grande utilité durant ma période d'apprentissage. Merci pour votre aide.

Je tiens à remercier l'ensemble du corps administratif et les enseignants de l'Ecole Hassania des Travaux Publics qui m'ont apporté collaboration et soutien durant mes études ainsi que tous les membres de la SG ATS pour leurs soutiens et leurs encouragements. Je remercie également les membres du jury, pour l'honneur qu'ils m'ont fait en acceptant d'examiner et juger mon travail.

Enfin, il m'est particulièrement agréable d'adresser mes remerciements à celles et ceux qui ont contribué à la réussite de ce projet que ce soit par une parole, un sourire ou un encouragement.

Résumé

Aujourd'hui, le recours au cloud est devenu incontournable pour l'entreprise qui souhaite suivre l'évolution technologique et gagner en productivité. Ce qui l'oblige à bien maitriser la gestion de l'architecture Cloud pour déployer ses différents services.

C'est dans ce sens que la Société Générale Africa Technogies & Services (SG ATS) a développé un service d'Auto Scaling pour assurer la fiabilité et la haute performance de ses services Client déployés sur le Cloud. Ce service se basait sur un fichier de configuration généré manuellement. Ce qui engendrait souvent soit une sous-estimation des ressources et donc une surcharge de services et un temps de réponse client insatisfaisant, soit une surestimation de ressources et donc un cout de déploiement très élevé sur le Cloud.

C'est dans ce cadre que s'inscrit mon projet de fin d'étude effectué à la SG ATS. Ma mission était de réaliser une application Desktop qui permet de simuler le fonctionnement du service de Scaling des différents services exposés sur le cloud Azure. Le but est d'améliorer l'efficacité du processus du Scaling et d'optimiser les charges de déploiement sur le Cloud.

Nous avons utilisé la méthode XP (eXtremme Programming) comme méthode de développement et de gestion de projet permettant des livraisons promptes et continues.

Nous nous sommes basés sur une architecture monolithique qui utilise .Net comme framework et C# comme langage de développement.

Nous avons ainsi réalisé un outil de simulation de service de Scaling qui génère trois types de résultats (sous forme de temps d'activité/exécution et de temps d'attente), à savoir les résultats des Machines de Service allouées sur le Cloud (Workers), les résultats de la file d'attente des requêtes Client et les résultats relatifs aux métriques Cloud.

Mots-clés :

Cloud, Métriques Cloud, Requêtes Client, Service Cloud, Service Scaling, Simulation, Workers**.**

Abstract

Today, the use of the cloud computing has become essential for companies wishing to keep up with the trend in technology and increase their productivity. This requires them to master the management of the Cloud architecture to start deploying their services in the cloud.

In that way Societe Generale Africa Technogies & Services (SG ATS) has developed an Auto Scaling service to ensure the reliability and high performance of its services deployed as cloud services. The auto scaling service is based on a configuration file generated manually. What often bring whether an underestimation of resources and therefore an overload of services and an unsatisfactory customer response time, or an overestimation of resources and a very high deployment cost.

In this context that my end of studies project carried out at SG ATS. My duty is to create a Desktop application which simulates the Scaling service function, which is ensuring a real time scaling operation for the various services exposed on the Azure cloud. The goal is to improve the efficiency of the Scaling process which lead to optimize the deployment costs.

I used the XP (eXtreme Programming) method as a development and project management method allowing prompt and continuous deliveries.

We adopt a monolithic architecture, and exploit the .Net framework and C# language to develop the various component of this solution.

We have thus developed a load simulation tool which generates three types of results:

- Activity & inactivity time for each machine allocated on the Cloud (Workers),
- Queue & execution time for each client request;
- Billing & performance metrics based on the giving scaling configuration;

Keywords :

Cloud, Cloud Metrics, Client requests, Cloud Service, Scaling Service, Load simulation, workers.

Table de matière

Les acronymes

Config	Configuration de fichier de scaling
CSV	Comma-separated values
DNS	Domain name service
Dry	Don’t repeat yourself
IaaS	Infrastructure as a service
IDE	Integrated development environment
Json	JavaScript Object Notation
PaaS	Platform as a service
RDWS	Research & Development Web Services
SaaS	Software as a service
SET	Software engineering technologies
SG Ats	Société générales Africa technologies & services
SG CIB	Société générales Corporation Investment Bank
UML	Unified Modeling Language
UX	User experience
VM	Virtual machine
Kiss	Keep it simple ,stupid
W3C	World Wide Web Consortium
XP	Extreme Programming
Yagani	you ain't gonna need it

Liste des figures

Liste des tableaux

Introduction générale

Les marchés financiers sont souvent considérés comme les piliers du capitalisme. Ils facilitent les échanges financiers, ainsi que l'achat et la vente des différents produits financiers.

En effet, la finance du marché est un domaine critique qui connaît une concurrence massive entre les différentes entités du marché. L'innovation s'impose comme un facteur décisif, d'où la nécessité d'investir dans la recherche et le développement.

Dans ce sens, la filiale de Recherche & Développement de la Société Générale Africa Technologies & Services a été créée par le groupe Société Générale en début 2014. Cette filiale a pour mission le développement des librairies de Pricing, outils et logiciels utilisés par toutes les entités du groupe.

Ce projet effectué au sein de la Société Générale Africa Technologies & Services, émerge d'une volonté du département Recherche & Développement à améliorer les outils utilisés par les ingénieurs de l'organisme, précisément les équipes du département de la recherche et développement.

L'un des processus à améliorer et qui est indispensable à la qualité de déploiement des services Cloud est le processus du choix de configuration du service Scaling.

L'idée consiste en la réalisation d'un simulateur de service de scaling, qui se comportera comme un laboratoire de test du fichier de configurations qu'il faut valider et rendre opérationnel.

Le présent rapport détaille les différentes phases du déroulement du projet, il se compose de cinq chapitres.

Le premier chapitre consiste en une présentation du projet et de son contexte. Il présente d'abord l'organisme d'accueil, suivi de la description du projet et de sa planification.

Le deuxième chapitre vient mettre le point sur l'analyse de l'existant, ses limitations et la solution proposée, suivis par un premier repérage des besoins fonctionnels et techniques, et par une modélisation dynamique du contexte.

Une présentation des besoins fonctionnels et non fonctionnels sera donnée dans le troisième chapitre.

La phase Analyse et Conception sera décrite dans le quatrième chapitre, appuyée par la présentation des différentes modèles statiques et dynamiques de la solution proposée. Le dernier chapitre décrit la phase Réalisation et Mise en œuvre du projet.

À la fin de ce mémoire vient la conclusion générale sous forme de bilan du projet s'ouvrant sur des perspectives, suivie d'une webographie et d'annexes en complément afin de mieux appréhender le contenu.

Chapitre 1 : Contexte général du projet

Ce chapitre met l'accent sur l'environnement de travail. Une première partie sera consacrée à la présentation de l'organisme d'accueil. La seconde partie a pour but principal d'exposer le cadre général du projet ainsi que ses différents objectifs. Quant à la dernière partie, elle présente la démarche suivie pour la gestion du projet.

1. Présentation de l'organisme d'accueil

1.1 Groupe Société Générale

Société Générale est l'un des tout premiers groupes européens de services financiers. S'appuyant sur un modèle diversifié de banque universelle, le groupe allie solidité financière et stratégie de croissance durable. L'ambition : être la banque relationnelle, référente sur ses marchés, proche de ses clients, choisie pour la qualité et l'engagement de ses équipes.

Figure 1: Logo du groupe

Le Groupe a été créé en 1864 pour favoriser le développement du commerce et de l'industrie française. Il accompagne aujourd'hui 31 millions de clients dans 67 pays avec 149.000 collaborateurs dont 61% exercent à l'international (hors France). La figure 2 représente implantation géographique de la Société Générale.

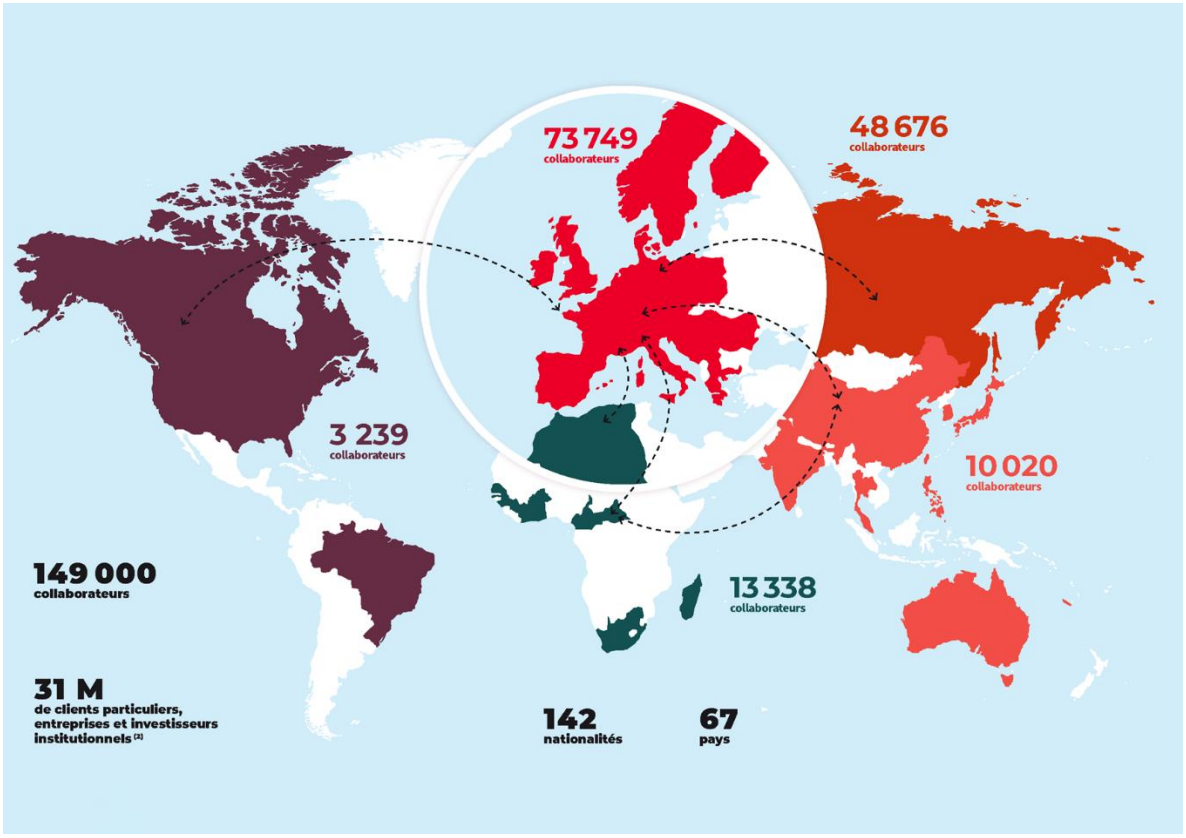

Figure 2: Implantation géographique de la Société Générale

1.2. Société Générale Corporate & Investment Banking (SG CIB)

SG CIB est la banque d'investissement du groupe Société Générale, elle est présente dans plus de 38 pays et 11000 personnes y travaillent. Elle accompagne les entrepreneurs, les grandes entreprises, les instituts financiers, les gouvernements, et les investisseurs à établir des liens sur le marché des capitaux, et leur fournit des solutions de financement, d'investissement et de gestion des risques.

1.3. Société Générale Africa Technologies & Services

Société Générale Africa Technologies & Services (SG ATS), filiale du groupe Société Générale Corporate & Investment Banking, regroupe les services Recherche & Développement des activités de marché de la Banque de Financement et d'Investissement. Basée à Casablanca, Société Générale Africa Technologies & Services a ouvert ses portes début 2014. Les nouvelles réglementations bancaires ont introduit de nouveaux indicateurs de mesure de risque (VaR sur CVA …) et ont renforcé les indicateurs existants (VaR, EEPE). Pour répondre à ces évolutions, les salles de marché de Société Générale ont besoin d'outils de Pricing et de calculateurs de risques plus sophistiqués, plus performants (notamment grâce à digitalisation de ces outils) et plus précis en réponse aux évolutions réglementaires. Ces efforts de Recherche & Développement (R&D) sont ainsi assurés par les équipes de Société Générale Africa Technologies & Services. Les équipes travaillent en étroite collaboration avec les équipes R&D à Paris, Londres, New-York et Hong-Kong ainsi qu'avec le Front Office afin de concevoir et développer de nouveaux outils toujours plus performants et d'assurer l'intégration et le support des librairies financières utilisées à la formation des prix de produits financiers.

La figure suivante illustre l'organigramme de SG ATS :

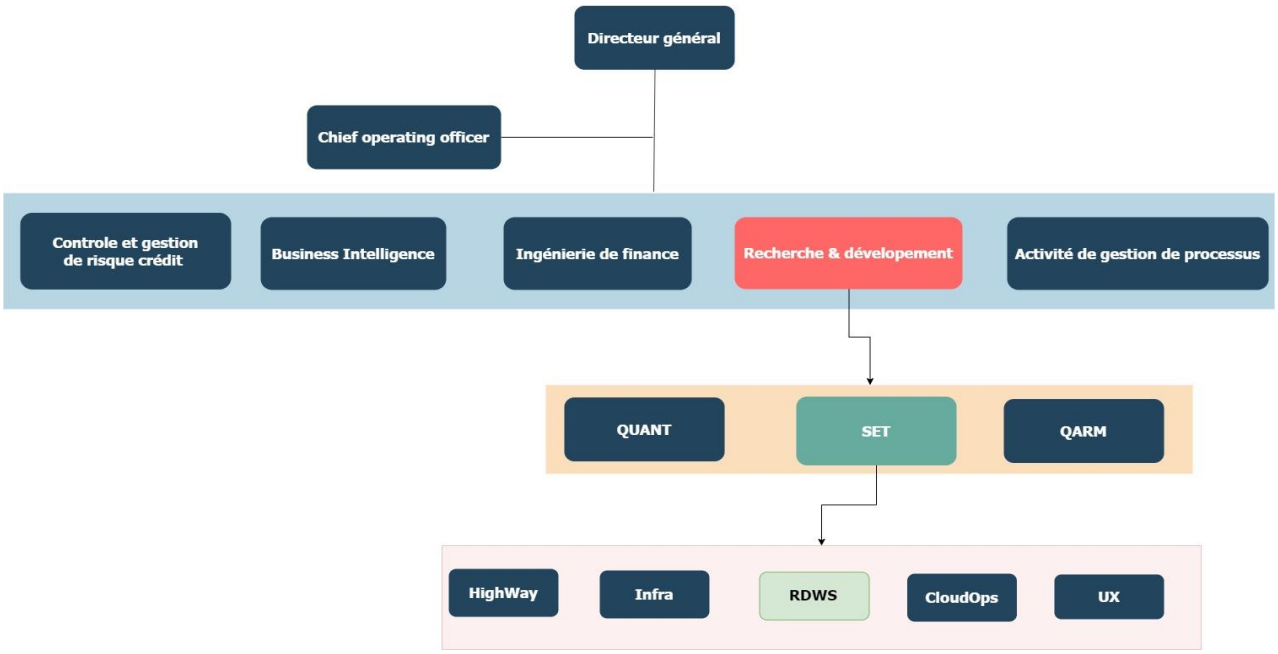

Figure 3: Organigramme de SG ATS

L'organigramme de SG ATS représenté par la figure 3 est encore plus riche mais nous détaillerons juste le pôle Recherche et Développement.

Le pôle R&D est constitué des équipes suivantes :

- ✓ QUANT : a pour mission de développer les librairies mathématiques de calcul des prix.
- ✓ QARM : l'équipe en charge de tests et de validation des produits développés par les autres équipes.
- ✓ SET : l'équipe en charge de la création des outils de trading et de la mise en œuvre effective des librairies de calcul dans un contexte industriel.

Le projet a été développé au sein de l'équipe RDWS (Research et development web services) faisant partie de l'équipe SET, qui a pour missions de développer et maintenir des solutions MicroServices destinées à être déployé sur les plateformes Cloud Azure.

2. Présentation du projet

Une bonne définition du cadre de projet s'avère particulièrement importante dans le déroulement du projet et pour atteindre les résultats escomptés. Cette section comporte la description du projet ainsi que ses objectifs.

2.1 Contexte du projet

L'équipe RDWS développe et maintient des services utilisés par les équipes du département Recherche & Développement du groupe Société Générale. Ces services sont regroupés et déployés dans le cloud Microsoft Azure.

Afin de comprendre le contexte, il est jugé nécessaire de présenter le contexte Cloud Computing et les services IaaS, PaaS, et SaaS.

a) Cloud Computing contexte :

D'après la définition du Microsoft Azure**, Le Cloud Computing** est la fourniture de services informatiques (notamment des serveurs de stockage, des serveurs de bases de données, la gestion du réseau) via Internet, dans le but d'offrir une innovation plus rapide, des ressources flexibles et des économies d'échelle. En règle générale, on paye uniquement les services cloud utilisés, en réduisant ainsi les coûts d'exploitation, et on gère l'infrastructure plus efficacement.

La figure 4 illustre les différentes branches du cloud computing :

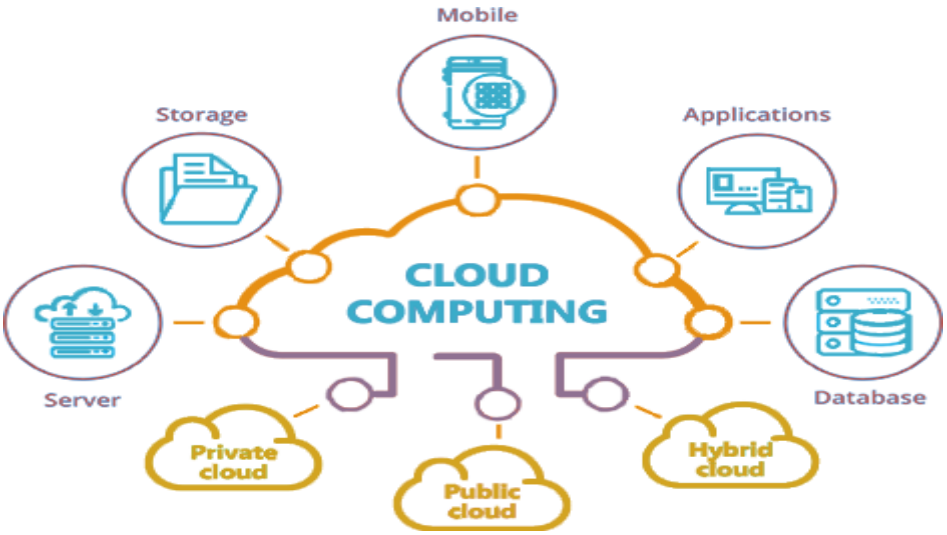

Figure 4 : La carte du Cloud Computing

Le cloud Computing offre plusieurs avantages, parmi ces avantages on trouve :

- ✓ Coût : Le cloud computing élimine la nécessité d'investir dans du matériel et des logiciels,
- ✓ La mise à l'échelle élastique : est un des avantages des services de cloud computing. En termes de cloud, cela veut dire qu'il est possible de mettre en œuvre la quantité nécessaire de ressources informatiques, par exemple plus ou moins de puissance de calcul, de stockage ou de bande passante, au moment où elles sont nécessaires, là où elles sont nécessaires.
- ✓ Performances : Les plus grands services de cloud Computing s'exécutent sur des machines très performantes en termes de ressources pour assurer la rapidité et l'efficacité.
- ✓ Sécurité : De nombreux fournisseurs de cloud offrent un vaste éventail de stratégies, technologies et contrôles qui renforcent globalement votre situation de sécurité, contribuant ainsi à protéger vos données, vos applications et votre infrastructure contre des menaces potentielles.
- ✓ Flexibilité : La plupart des services de cloud Computing sont fournis en libre-service et à la demande. D'énormes ressources de calcul peuvent donc être mises en œuvre en quelques minutes et en quelques clics, offrant ainsi aux entreprises un haut niveau de flexibilité et les dégageant de la pression liée à la planification de la capacité.

La figure 5 montre les principaux avantages du Cloud Computing :

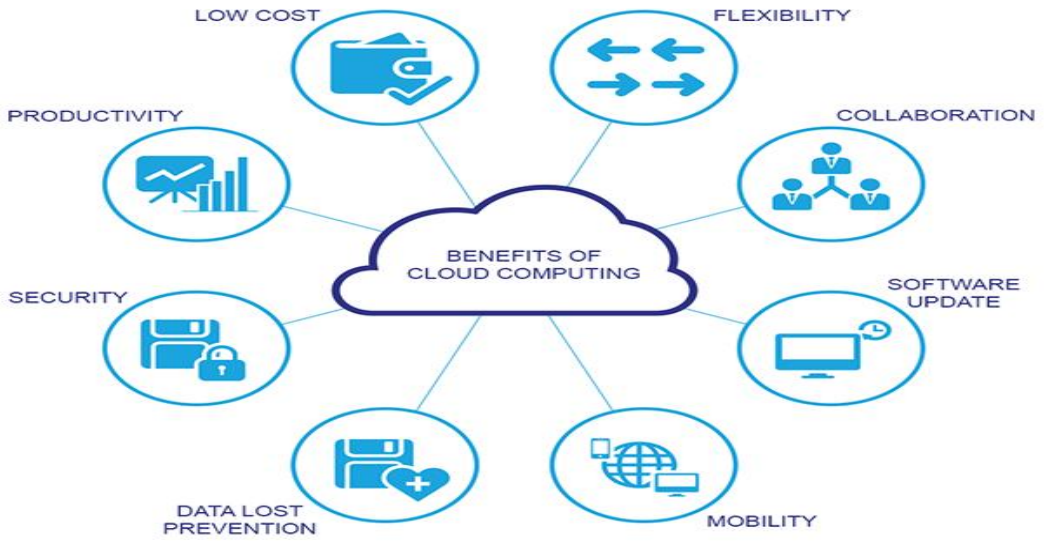

Figure 5: Les avantages du Cloud Computing

b) *Les modèles d'hébergement sur le Cloud*

D'après la W3C [World Wide Web Consortium] l'organisation mondiale de standardisation du Web, il existe trois modèles d'hébergement sur le Cloud capables de répondre aux besoins spécifiques de chaque entreprise :

- **IaaS : Infrastructure as a Service** connus sous le nom d'infrastructure en tant que service (IaaS), permet aux entreprises de gérer ses applications, des données, du temps d'exécution, du middleware et des systèmes d'exploitation. En outre, la solution IaaS vous délivre et gère la virtualisation, les serveurs, les disques durs, le stockage et les réseaux. Les services d'infrastructure Cloud, sont des modèles en libre-service pour accéder, surveiller et gérer des infrastructures de centres de données distantes, telles que des services de calcul (virtualisés ou non), de stockage et de réseaux. La solution IaaS permet aux entreprises de payer uniquement ce qu'elles consomment (pay-as-you-use). Ceci permet d'éviter l'achat de matériel physique et de payer plus que ce que l'entreprise ne consomme réellement.

Les utilisateurs obtiennent avec le IaaS une infrastructure sur laquelle ils peuvent installer tous types de plates-formes. Ils sont responsables de la mise à jour des couches supérieures (PaaS). IaaS permet au client un développement beaucoup plus large que le PaaS.

- **PaaS : Platforme as Sevice**, connus sous le nom de plate-forme en tant que service (PaaS) fournit tous les composants et l'infrastructure Cloud pour développer, déployer et gérer des applications. Les entreprises ne se concentrent ici que sur les applications. Les données, le temps d'exécution, les middlewares et les systèmes d'exploitation sont ici gérés par le fournisseur. Ainsi, l'avantage du PaaS pour les développeurs est la personnalisation des environnements à leur souhait. Le service PaaS rend le développement, le test et le déploiement d'applications rapides, simples et rentables. Les entreprises peuvent vraiment se concentrer sur le côté commercial, l'évolutivité et

le développement de l'application de leur produit ou service. Les applications utilisant PaaS bénéficient des caractéristiques Cloud telles que l'activation de SaaS, la haute disponibilité, la multi-location, l'évolutivité (Scaling) et plus encore.

Il reste à mentionner que Paas est une sorte d'abstraction pour IaaS, car au lieu d'allouer une machine et installer les applications dont on a besoin, le PaaS offre l'architecture et les applications en même temps, pour but de simplifier la tâche de virtualisation et déploiement aux entreprises.

- **SaaS :** Le Mode SaaS **Software as a Service** ou logiciel en tant que service permet à l'entreprise de profiter d'un logiciel via Cloud plutôt que d'avoir à l'installer sur ses propres serveurs ou ordinateurs, comme ça l'entreprise paye ce qu'elle consomme en terme des heures d'utilisations d'un produit donné au lieu de payer une licence mensuelle ou annuelle.

La figure 6 illustre l'architecture des modèles d'hébergement sur le cloud :

Figure 6: Les modèles d'hébergement sur le Cloud

c) *L'architecture d'un service Cloud Azure*

Le portail Azure permet de créer et de déployer des applications Web sous forme de services Cloud.

Un service Cloud s'étale sur un ensemble de réplicas ou bien d'instances, chaque instance de service est installée sur une machine virtuelle (**Worker**).

La figure suivante illustre l'architecture d'un service Cloud Azure :

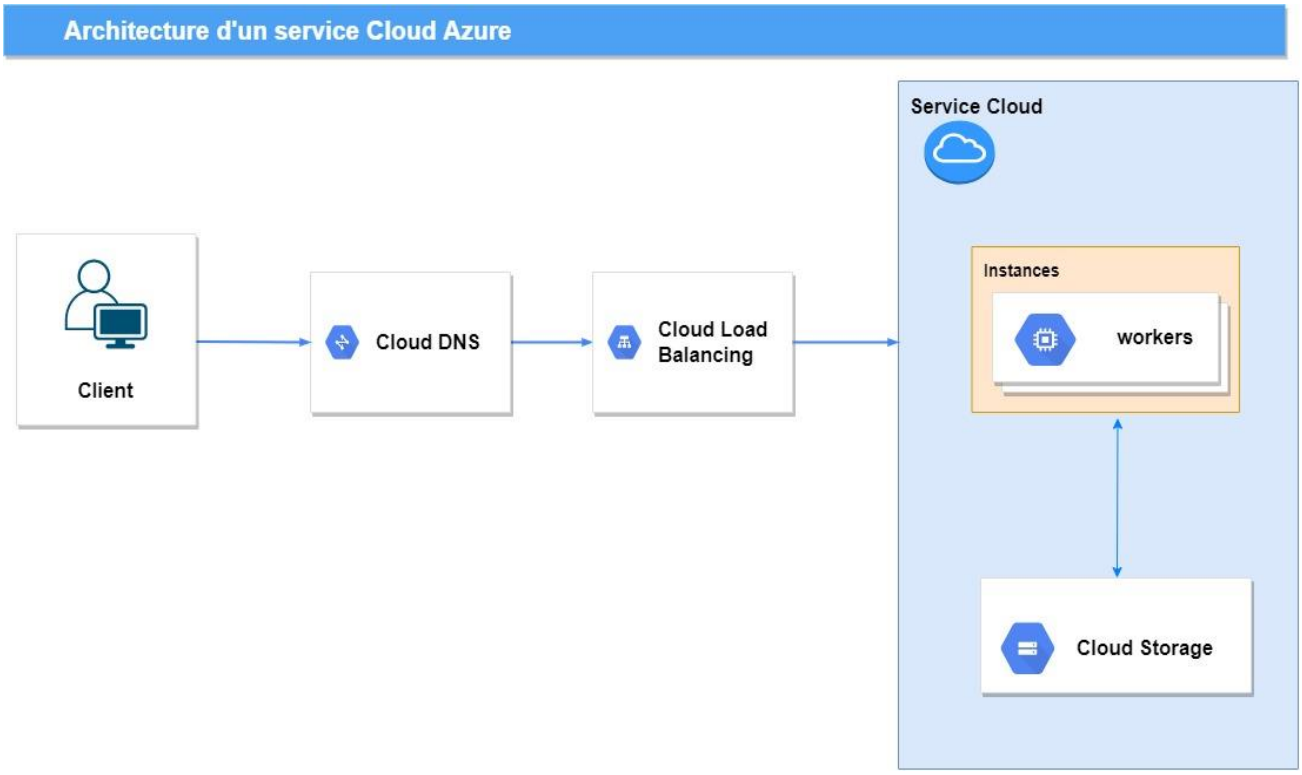

Figure 7: Architecture d'un service Cloud Azure

Pour une gestion optimale des workers, le fournisseur du cloud met à disposition de ses clients un tas de services, à savoir :

- **Cloud DNS** : c'est pour la redirection des requêtes selon le type de service demandé ;
- **Cloud Load Balancing** : C'est le serveur de gestion de charge qui permet de gérer la charge équitablement entre les instances d'un service ;
- **Cloud Storage** : Un répertoire de stockage partagé entre les instances d'un service ;
- **Workers** : Des machines virtuelles qui exécutent une instance d'un service.

d) L'Auto Scaling

L'auto scaling est une méthode qui permet d'avoir le nombre de machine nécessaires pour gérer la charge d'une application. Il permet de rajouter des ressources pour réagir face à une situation de surcharge, mais aussi d'en supprimer en cas d'excès de machines inactives.

Techniquement parlant, on commence par un nombre minimal de machines, et on fixe un nombre maximal de machines, et selon la charge le nombre de machines virtuelles augmente ou diminue automatiquement.

La figure Suivante illustre le concept de scaling :

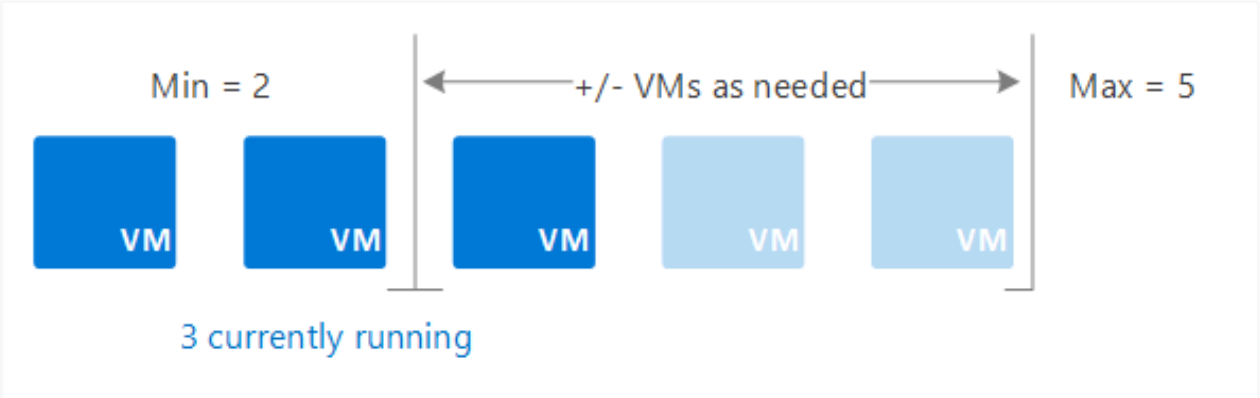

Figure 8: Concept d'auto Scaling

2.2 L'existant

Pour assurer la fiabilité et la haute performance de ses services Cloud, SG ATS et précisément l'équipe RDWS a développé un service d'Auto Scaling qui s'occupe de la tache de la mise à l'échelle des services Cloud.

Le service de Scaling prend en paramètre un fichier de configuration composé par des unités de configurations applicables dans des périodes bien précises au cours la semaine.

Une unité de configuration est composée d'un ensemble de paramètres :

- ✓ Le nombre minimal de machines à créer,
- ✓ Le nombre maximal de machines à créer,
- ✓ Le pourcentage minimal de machines inactives (machines en attente de requêtes),
- ✓ Le pas de Scaling qui signifie le nombre de machines à éteindre en cas d'excès de machines inactive,
- ✓ Days_of_week ou les jours d'application de l'unité configuration,
- ✓ Utc_start_time ou l'heure de début d'application de l'unité configuration,
- ✓ Utc_end_time ou l'heure de fin de validité de l'unité de configuration.

Le service de scaling s'appuie sur un autre service qui s'appelle Service-Watcher, ce dernier fait des captures d'état (workers HeartBeat) sur les machines virtuelles d'une façon périodique, et envoie l'état de chaque machine au service d'auto Scaling.

En se basant sur le fichier de configuration et l'état des machines virtuelles fournis par le service Watcher, le service Scaling s'exécute périodiquement en temps réel pour calculer le nombre optimal d'instances pour un service.

Le service de scaling communique ces résultats à un service Azure cloud **Zscaler**, ce dernier offre un accès privé pour contrôler son architecture cloud en temps réel.

La figure 10 illustre l'architecture de fonctionnement du service d'auto Scaling :

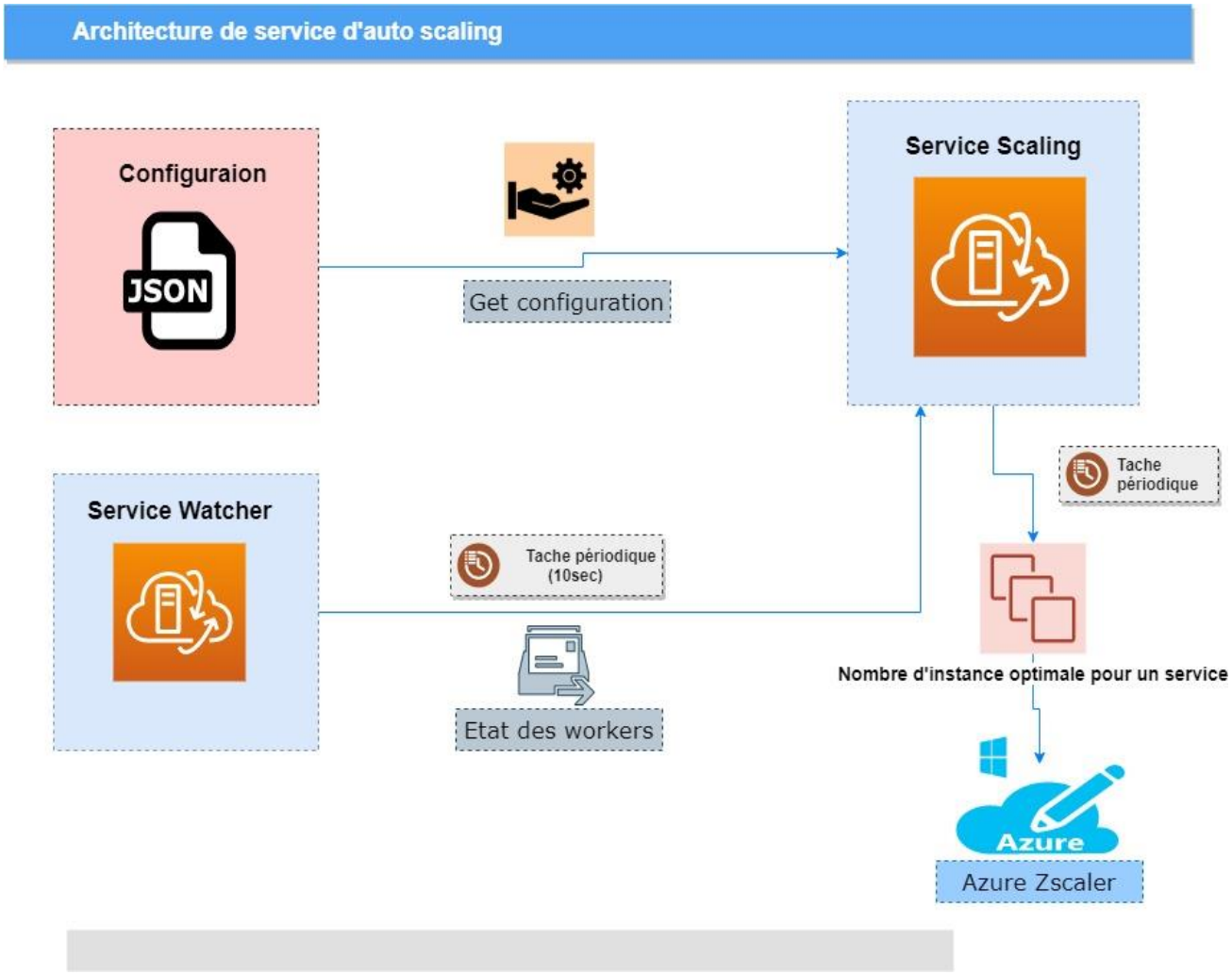

Figure 9: Architecture du service d'auto scaling

2.3 Problématique

Le service d'auto scaling existant se base sur un choix manuel des configurations. Ce choix manuel fait par l'équipe CloudOps, engendre les problèmes suivants :

- ✓ Une sous-estimation de la charge des requêtes due au fait de donner une configuration minimale en termes de ressources, ce qui génère une surcharge de services et par conséquent un temps de réponse non satisfaisant pour le client.
- ✓ Une surestimation de la charge due à une configuration très gourmande en termes de ressources, ce qui génère un coût de déploiement sur le cloud très élevé.

2.4 Objectifs du projet

Le but du stage est la création d'un outil de simulation qui permet de simuler le fonctionnement du service d'auto Scaling.

Étant donnés un fichier de configurations et une charge virtuelle de requêtes Client (collectées à partir de l'historique d'un service Cloud), cet outil va pouvoir ressortir des documents qui présentent :

- L'état de la queue des requêtes Client au cours de la simulation,
- L'état des machines virtuelles (ou workers) qui hébergent chacune une instance de service répondant à une requête client au cours de la simulation,
- Les facteurs de qualité ou précisément ce qu'on appelle les Métriques Cloud relatives au déploiement d'un service Cloud.

En se basant sur ces documents, on peut mesurer la qualité et la fiabilité de déploiement d'un service et par la suite évaluer la qualité du fichier de configurations qui a engendré ces résultats.

A ce titre les éléments suivants doivent être assurés :

- ✓ Bien comprendre la problématique pour répondre aux besoins ;
- ✓ Bien analyser l'existant ;
- ✓ Proposer, concevoir et développer la solution d'une façon agile ;
- ✓ Agir avec rigueur afin de fournir un outil de qualité ;
- ✓ Mesurer sa performance une fois que la solution est mise en production.

2.5 Enjeux du projet :

Pour évaluer la qualité d'un fichier de configurations, on se trouve face à deux enjeux principaux :

- ✓ **Réduire le coût en termes de ressources matérielles** : Une configuration adéquate est censée réduire le nombre de machines allouées sur le cloud et par conséquent optimiser les charges.
- ✓ **Minimiser le temps d'attente d'une requête Client** : Ceci permet d'avoir un nombre suffisant de machines allouées sur le cloud capables de supporter la charge des requêtes Client dans les périodes les plus actives en termes de transactions.

3. Conduite de projet

3.1 Méthodologie de travail :

La méthode **XP** (eXtreme Programming) est la méthode adoptée pour la gestion de projet.

La figure 11 montrent le cycle de développement recommandé par la méthode XP :

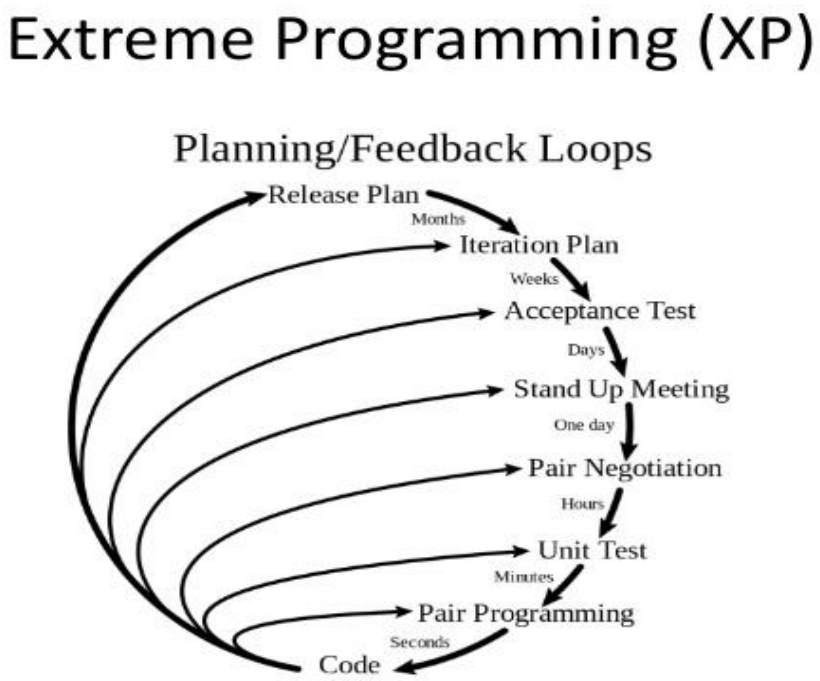

Figure 10: Cycle de la méthode XP

D'après le livre [Extreme Programming Explained] publié en 1999 par Kent Beck le créateur et l'inventeur de la méthode XP, il expliqua dans son livre que la méthode XP appartient à la famille des méthodes agiles plus particulièrement orientée sur l'aspect réalisation d'une application, sans pour autant négliger l'aspect gestion de projet. XP est adapté aux équipes réduites avec des besoins changeants. Les principes de la méthode XP ne sont pas nouveaux puisqu'il s'agit de ceux des méthodes Agiles. La différence et l'originalité résident dans le fait qu'ils sont poussés à l'extrême : Simplicité, Feedback, Communication.
La méthode eXtreme Programming s'appuie sur :

- ✓ Des petits livrables.
- ✓ Une conception simple.
- ✓ Des tests unitaires.
- ✓ La revue du code.
- ✓ Du Refactoring (ou remaniement du projet).

La caractérisation de la méthode XP dans ce projet a été faite en suivant les pratiques telles que :

- ✓ **La revue du code** : pour chaque partie développée, une revue du code est prévue afin de valider l'incrément, et l'intégrer dans le répertoire distant. Cette revue a pour but d'assurer que le code respecte toutes les règles de qualité et les bonnes pratiques (YAGNI, KISS, DRY, SOLID…) imposées par l'équipe.
- ✓ **Les tests unitaires** : Les tests unitaires de chaque fonctionnalité et une couverture du code à 100% sont exigés par l'équipe.
- ✓ **Le *refactoring* du code** : Consiste à modifier le code source de façon à en améliorer la structure, sans que cela modifie son comportement fonctionnel.

3.2 Planification :

La planification d'un projet est une étape incontournable dans sa conduite. Nous avons donc établi un planning prévisionnel qui décrit le déroulement de notre projet deux fois à cause de la pandémie du Covid 19. Devant cette situation, nous étions obligés de découper notre stage en :

- Une période de développement et de tests directs dans les locaux de la SG ATS ;
- Une période de confinement avec interruption du stage entre le 20 mars 2020 et le 04 mai 2020 ;
- Une période de télétravail et reprise de stage.

Les deux diagrammes de Gantt suivants permettent de voir le planning de notre projet avant et après l'interruption du stage à cause de la pandémie Covid 19 :

Avant le 20 mars (date de confinement)

TASK NAME	START DATE	END DATE	WORK DAYS
Démarrage du projet	3/2/2020	7/2/2020	5 jours
Docuementation sur l'architecture Microservice Formation C#, Solid, UnitTesting, Git ... Elaboration du cahier de charge			
Construction du projet	10/2/2020	30/06/2020	74 jours
Etude préliminiare	10/2/2020	21/2/2020	10 jours
Documentation sur le fonctionnement de service scaling Documentation sur le fonctionnement de service watcher Documentation sur l'objet de configuration Json Documentation sur le fonctionnement d'un worker cloud Benchmarking Métriques cloud Documentation sur la tarification d'Azure workers Etude de l'existant (limitations & solution proposée) Identification des besions fonctionnels et techniques			
Spécifications générales	24/2/2020	27/2/2020	4jours
Identification des besoins Spécifications fonctionnelles Spécifications Techniques			
Plannigng de dévéloppement des iterations	28/02/2020	19/06/2020	60 jours
Itération 1 : Récupération des données & implémentation des objets de base	28/2/2020	5/3/2020	5jours
Extraction des requêtes clients à partir d'un fichier csv Extraire une unité de configuration à partir d'un fichier Json Conception des interfaces : Configuration, ClientRequest, et Worker Implémentation des interfaces Les tests unitaires			
Itération 2 : Implémentation du simulaterQueue & simulaterWorkers	6/3/2020	12/3/2020	5jours
Analyse des événements de la queue (enqueue, dequeue,...) Analyse des fonctionnalités de CloudWorkers (addWorker, shutDownWorker, AffectRequest...) Conception de l'interface SimulaterQueue & SimulaterWorkers Implémenter l'interface SimulaterCloud & SimulaterWorkers Les tests unitaires			
Itération 3 : Implémenter le ScalerSimulater	13/3/2020	19/3/2020	5jours
Analyse des fonctionnalités de base de service de scaling (ScaleUp,ScaleDown, GetNextScalingDate ...) Conception de l'objet ScalerSimulater Implémenter l'interface ScalerSimulater Les tests unitaires			

Figure 11: Diagramme de Gantt partie 1

Reprise de stage (4/5/2020)

Itération 4 : Implémenter Simulator & Orchestrator & les tests d'intégration	4/5/2020	15/5/2020	10jours
Conception de l'objet Orchestrator (GetNextEvent...)			
Conception de l'objet simulateur (StartSimulation,EndSimulation)			
Implémenter l'objet DateTimeSimulater			
Gérer les dépendances du temps entre les différentes interfaces (Injecter la dépendance du temps)			
Les tests unitaires			
Les tests d'intégration			
Itération 5 : Extendre l'objet de configuration : IConfigurationHolderHolder	18/5/2020	22/5/2020	5jours
Extension de l'objet configuration pour couvrir les jours de la semaine			
Conception de l'interface IConfigurationHolder qui représente une configuration hebdomadaire			
Implémenter l'interface			
Intégrer IConfigurationHlder à la solution existante			
les tests unitaires			
les tests d'integration			
Itération 6 : Implémenter les Métriques	25/6/2020	5/6/2020	10jours
Conception de l'interface MetricCloud (getIdleTime,GetCost...)			
Implémentation de l'interface MetricCloud			
Génération de fichier pour les metrics Cloud : metrics.csv			
Les tests unitaires			
Les tests d'intégration			
Itération 7 :Implémentation du Simulation ResultPrinter	8/6/2020	19/6/2020	10jours
Conception de l'interface SimulationResultPrinter			
Implémentation de SimulationResultPrinter			
Générer le fichier de résultats des workers : workers_result.csv			
Générer un fichier de résultats des requestes : requests_Result.csv			
Générer un fichier de résultats des requestes : rmetrics_Result.csv			
Afficher les résultats de la simulation			
Documentation	22/6/2020	8/7/2020	13 jours
Rapport de stage détaillé			
Manuel d'utilisation			

Figure 12: Diagramme de Gantt partie 2

3.3 Ressources :

Le suivi et l'organisation du projet sont illustrés sur le tableau suivant :

	Nom	Rôle
Comité de suivi	Mme. Malika Addou	Encadrante EHTP
	M. Tarik Douiyeh	Encadrant SG Ats
	M. Anas Bouaoud	Team Leader RDWS
Maitrise d'œuvre	M. Omar Zouaid	Elève ingénieur à EHTP

Tableau 1 : Suivi et l'organisation du projet

4. Conclusion

Le premier chapitre présentait un point de départ pour l'élaboration du projet dans la mesure où il définit son contexte général à savoir l'organisme d'accueil SG Ats, les objectifs du projet et la démarche suivie pour sa réalisation.

Dans la partie suivante, on présente les spécifications des besoins fonctionnels et techniques de la solution attendue.

Chapitre 2 : Étude Préliminaire

L'étude préliminaire est la première étape de notre processus de développement. Elle consiste à effectuer un premier repérage des besoins fonctionnels et opérationnels. Dans cette phase je vais mettre l'accent sur l'étude de l'existant pour recenser par la suite les besoins fonctionnels et ceux techniques.

1. Étude de l'existant

1.1 Génération du fichier de configuration et les Pull Client

Le fichier de configuration est généré par l'équipe cloudOps. En effet les membres de cette équipe construisent le fichier de configuration du service scaling en se basant sur l'historique du trafic des requêtes pour chaque service.

Le fichier de configuration est un fichier Json composé de plusieurs objets de configurations, chaque objet est dédié à un Pull Client (Groupe de client).

La figure 14, illustre la composition d'un fichier de configuration :

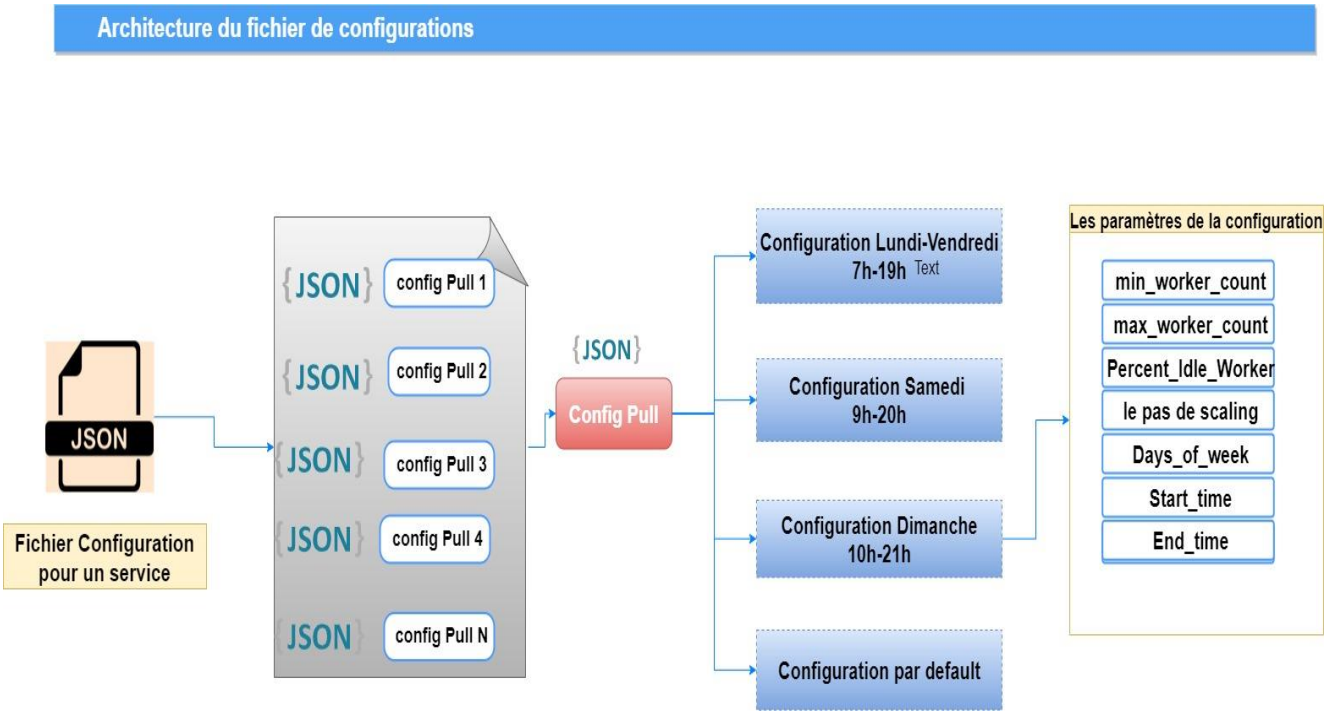

Figure 13: Architecture de fichier de configuration d'un service

Comme cité précédemment, les clients de SG ATS sont regroupés dans des Pull Clients, chaque groupe de clients est caractérisé par un nombre de requêtes par heure, et donc un nombre de machines allouées sur le Cloud suffisantes pour supporter la charge des requêtes.

La figure suivante montre la répartition des instances d'un service A sur des Pull Workers, chaque Pull worker correspond à un Pull client :

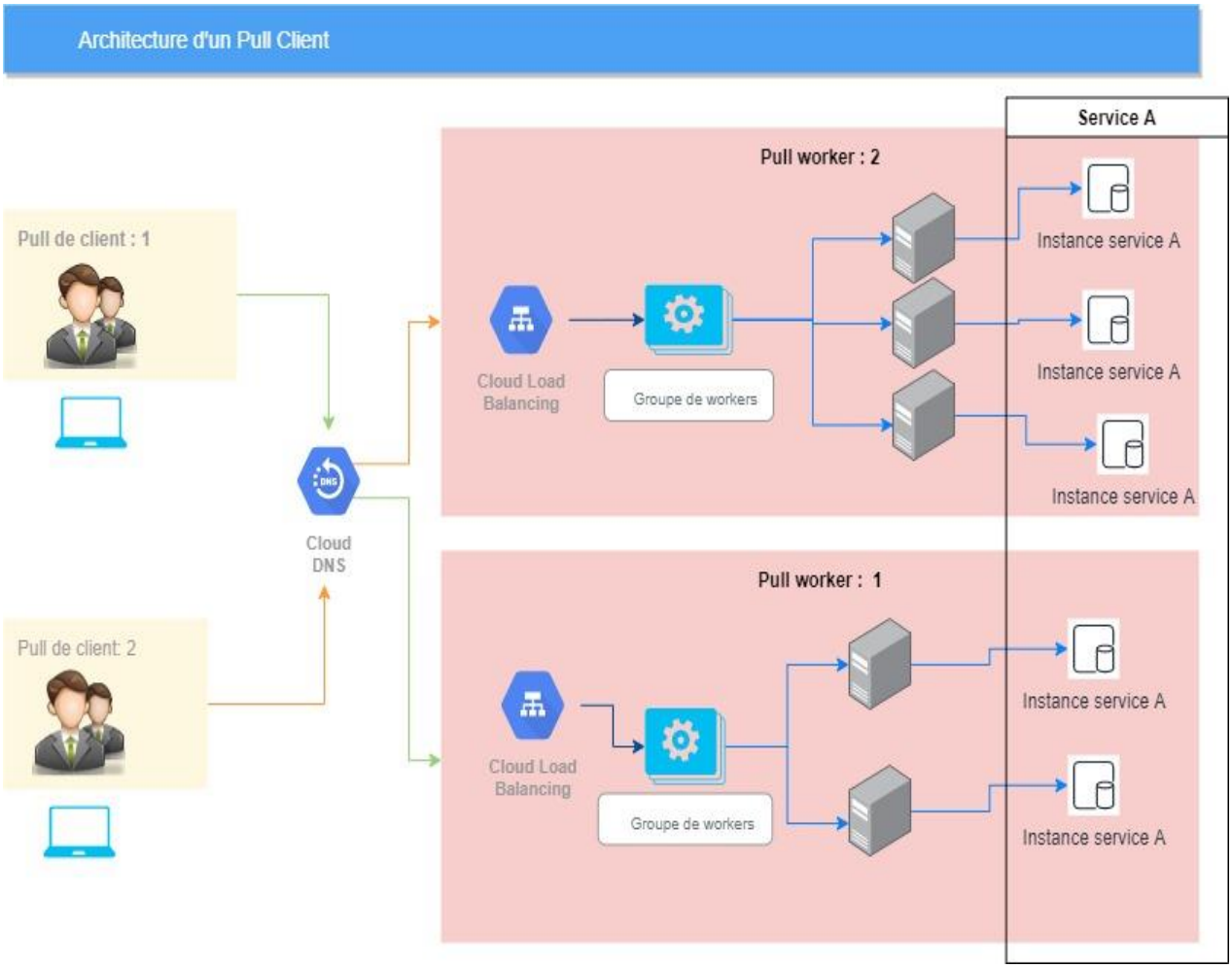

Figure 14: Architecture Pull Client

1.2. Limitations de l'existant

Dans un contexte professionnel on ne peut pas faire confiance aux choix humains, il est jugé nécessaire d'avoir un processus automatisé qui peut surpasser et éviter les erreurs censées être générées par l'être humain.

Dans ce sens, et en parlant du processus de génération de configuration, on remarque qu'il manque une phase de test et validation de la configuration avant qu'elle soit mise en production.

1.3. Solution proposée

La solution proposée consiste à créer une application Desktop de simulation de charge du service de scaling. Cette application se comporte comme étant un laboratoire de test pour évaluer les résultats d'une configuration, avant sa mise en production.

Si on sait mesurer la satisfaction du Client Cloud et calculer le coût du Fournisseur Cloud, on peut améliorer la qualité de déploiement des services sur le cloud. C'est lié en effet au nombre de workers alloués et à leur temps d'attente d'une part, d'autre part c'est lié aux temps d'attentes des requêtes Client sur la file d'attente. Le simulateur se chargera ainsi de fournir :

- L'état des Workers au cours de la simulation ;
- Les résultats d'exécution de chaque requête ;
- Les métriques ou facteurs de qualité du fichier de configuration.

2. Recueil des besoins

2.1 Besoins fonctionnels

Ce projet de fin d'étude entre dans le cadre de la digitalisation du processus de génération de configuration du service scaling. Son objectif est de réaliser un simulateur de service de Scaling qui offre un environnement de test pour les configurations avant leur mise en production. Les principales fonctionnalités peuvent être résumées dans ce qui suit :

Etat des Workers :

L'outil doit pouvoir ressortir en premier lieu un fichier csv qui présente l'Etat des Workers au cours de la simulation, à savoir le temps d'inactivité et d'activité de chaque Worker.

Etat de la queue des requêtes :

Les requêtes Client sont placées dans une file d'attente (ou queue). L'outil doit générer un autre fichier csv qui présente le temps d'attente et le temps d'exécution total de chaque requête dans la file.

Les métriques Cloud :

L'outil doit pouvoir ressortir les facteurs de qualité d'une configuration donnée. On distingue deux types de métriques :

- Les métriques de facturation : Ce sont des métriques relatives au montant payé pour le fournisseur cloud ;
- Les métriques de performance : Ce sont des métriques relatives à la qualité de déploiement du service Cloud, et qui influence directement la satisfaction du Client ;

a) Les métriques de facturation :

- **Workers Cost :**

  Le montant dû au fournisseur cloud pour l'allocation des Workers, cette métrique est calculée en multipliant le temps d'exploitation des workers par le prix d'allocation par minute ;

- **Average Workers effeciency :**
  La moyenne d'efficacité des workers (l'efficacité d'un worker est calculée en divisant le temps d'attente par le temps de travail d'un worker) ;

- **Workers Total Idle Time :**
  Le temps d'attente total des workers.

- **Additional Workers Cost :**
  Le montant dû au fournisseur cloud pour l'allocation des workers durant le temps d'attente, cette métrique est calculée en multipliant le temps d'inactivité total des workers par le prix d'allocation par minute.

b) Les métriques de performances :

- **Workers down Time :**
  Le temps durant lequel le service était indisponible (aucun worker disponible).

- **Workers Unavailability Percentage :**
  Le pourcentage du temps dans lequel les Workers étaient indisponibles par rapport au temps de la simulation.

- **Average Queue Time :**
  La moyenne du temps d'attente des requêtes dans la file d'attente.

- **Max Queue Time :**
  Le temps d'attente maximal d'une requête dans la file d'attente.

- **Percentage of Request with Zero Queue Time :**
  Percentage des requêtes qui sont passées à la phase d'exécution sans attendre dans la file.

2.2 Exigences techniques

Les exigences techniques sont des exigences qui ne concernent pas directement les services spécifiques fournis par le système. Ils peuvent concerner des propriétés et des contraintes remplies par le système dans son intégralité.

Voici les différentes spécifications techniques du système de simulation :

- **Performance :**

  La solution proposée doit être performante, avec un temps d'exécution négligeable devant celui des outils simulés (quelques millisecondes).

- **Maintenabilité :**

  - ✓ Le projet est de type évolutif, à chaque moment on peut intégrer de nouvelles idées et fonctionnalités, et pour cela il faut éviter le fort couplage entre les différentes composantes de la solution.
  - ✓ Les tests unitaires doivent couvrir la totalité du code source avec un taux de couverture qui tend vers 100%, ce qui va simplifier la maintenabilité.
  - ✓ Les tests unitaires doivent aussi être fait d'une façon très stricte, afin qu'on puisse tester tous les comportements possibles de toutes les classes.
  - ✓ Il faut respecter les principes SOLID durant la phase de développement.
  - ✓ La barre des objectifs est très haute, il se peut qu'on n'arrive pas à les réaliser tous, et donc pour cela nous devons assurer l'évolution de la solution soit pour garder la performance ou l'améliorer.

- **Evolution**

  - ✓ La barre des objectifs est très haute, il se peut qu'on n'arrive pas à les réaliser tous, et donc pour cela nous devons assurer l'évolution de la solution, soit pour garder la performance ou l'améliorer ou bien ajouter d'autres fonctionnalités.

3. Approches de simulation

1.3 Etude comparative des approches de simulation existantes

Un logiciel de simulation est basé sur le processus de modélisation d'un phénomène réel avec un ensemble de formules mathématiques. Il s'agit essentiellement d'un programme qui permet à l'utilisateur d'observer une opération par simulation sans réellement effectuer cette opération.

Il existe deux types de simulation :

- **La simulation continue** qui fait référence à un modèle informatique d'un système physique contenu dans le temps, à savoirs la chaleur, la pression etc.
- **Une simulation d'événements discrète** qui modélise le fonctionnement d'un système comme une séquence discrète d'événements dans le temps.

Le système simulé (service de Scaling) dans ce projet est un système d'évènements discrètes et dénombrables, c'est pour cela qu'on le modélisera par un simulateur d'évènements discrète.

Pour réaliser une simulation d'évènements discrète, on trouve qu'il y a deux approches possibles :

- Progression temporelle à incrément fixe ;
- Progression temporelle à incrément variable ;

a) Progression temporelle à incrément fixe

Dans la progression temporelle à incrément fixe, le temps est divisé en petites tranches et l'état du système est mis à jour en fonction de l'ensemble des événements qui se produisent dans la tranche du temps.

Cette méthode est conceptuellement simple à implémenter, mais elle n'est pas rapide. Le problème de rapidité est dû au fait que dans la méthode de progression à incrément fixe, il y aura surement des cycles traités au cours de la simulation qui ne contiennent aucun événement susceptible de se produire, d'où la nécessité d'une méthode plus rapide, optimale, et concise.

b) La progression temporelle à incrément variable

Dans La progression temporelle à incrément variable, Chaque événement se produit à un instant particulier et marque un changement d'état dans le système. Entre les événements consécutifs, aucun changement dans le système n'est supposé se produire, ainsi le temps de simulation peut directement passer au temps d'occurrence du prochain événement, qui est appelé progression temporelle du prochain événement.

Cette méthode présente une performance ultime, grâce à sa rapidité due au fait que le simulateur au lieu qu'il fasse des sauts fixes, il se déplace entre les événements qui se produisent au cours de la simulation.

Ainsi la méthode qui sera utilisée dans ce projet et la progression temporelle du prochain événement.

2.3 Les composants de la simulation discrète

Comme on a déjà cité, la méthode qui sera utilisée au niveau de notre simulateur est la progression temporelle du prochain événement. Cette méthode est composée de plusieurs éléments :

a) Horloge

La simulation doit garder une trace du temps de simulation actuel, dans toutes les unités de mesure qui conviennent au système modélisé. Dans les simulations à événements discrète, par opposition aux simulations continues, le temps «saute» parce que les

événements sont instantanés - l'horloge passe à l'heure de début de l'événement suivant à mesure que la simulation se poursuit.

b) État

Un état du système est un ensemble de variables qui capture les propriétés qui avancent du système à étudier et qui changent après chaque événement. Les variables d'état de notre simulateur sont les suivants :

- La liste des requêtes
- La Queue Cloud
- La liste des workers
- Le temps

c) Liste des événements

La simulation conserve au moins une liste d'événements de simulation. Ceci est parfois appelé l'ensemble d'événements en attente car il répertorie les événements en attente suite à un événement précédemment simulé mais qui n'ont pas encore été simulés eux-mêmes. Un événement est décrit par l'heure à laquelle il se produit et un type, indiquant le code qui sera utilisé pour simuler cet événement.

La liste des évènements de notre simulateur est la suivante :

- **Scaling Up** signifie la programmation de l'ajout des instances supplémentaires,
- **Scaling Down** signifie la suppression des instances worker,
- **Arrived Request** signifie l'arrivée d'une requête Client,
- **Dequeue time** signifie l'exécution d'une requête Client,
- **Launching Workers** signifie l'ajout de workers après l'installation des différents fichiers binaires nécessaires,
- **End of simulation** signifie la fin de la simulation.

d) Statistiques

La simulation garde généralement la trace des statistiques du système, qui quantifient les résultats intéressants.

Les résultats de la simulation sont les suivants :

- Etat des workers ;
- Etat de la queue ;
- Les métriques Cloud.

e) Ending condition

Parce que les événements sont amorcés, une simulation d'événements discrète pourrait théoriquement s'exécuter indéfiniment. Le concepteur de simulation doit donc décider de la fin de la simulation.

Les conditions d'arrêt sont les suivantes :

- Soit file d'attente vide et les workers inactifs ;
- Soit l'intervalle du temps inséré par l'utilisateur est écoulé.

4. Modélisation du contexte du système

3.1 Identification des acteurs du système

Selon la notation UML un acteur représente l'abstraction d'un rôle joué par des entités externes qui interagissent directement avec le système étudié. Dans ce qui suit, je présente les acteurs qui interagissent avec notre application.

Dans le cas de mon projet, j'ai un seul acteur il s'agit d'un membre de l'équipe CloudOps.

3.2 Diagramme de contexte

Le diagramme de contexte dynamique illustre le système comme objet central interagissant avec un acteur principal à travers un ensemble messages. L'objectif de cette représentation est de montrer une vue globale du système développé.

La figure 15 montre le diagramme de contexte de notre solution :

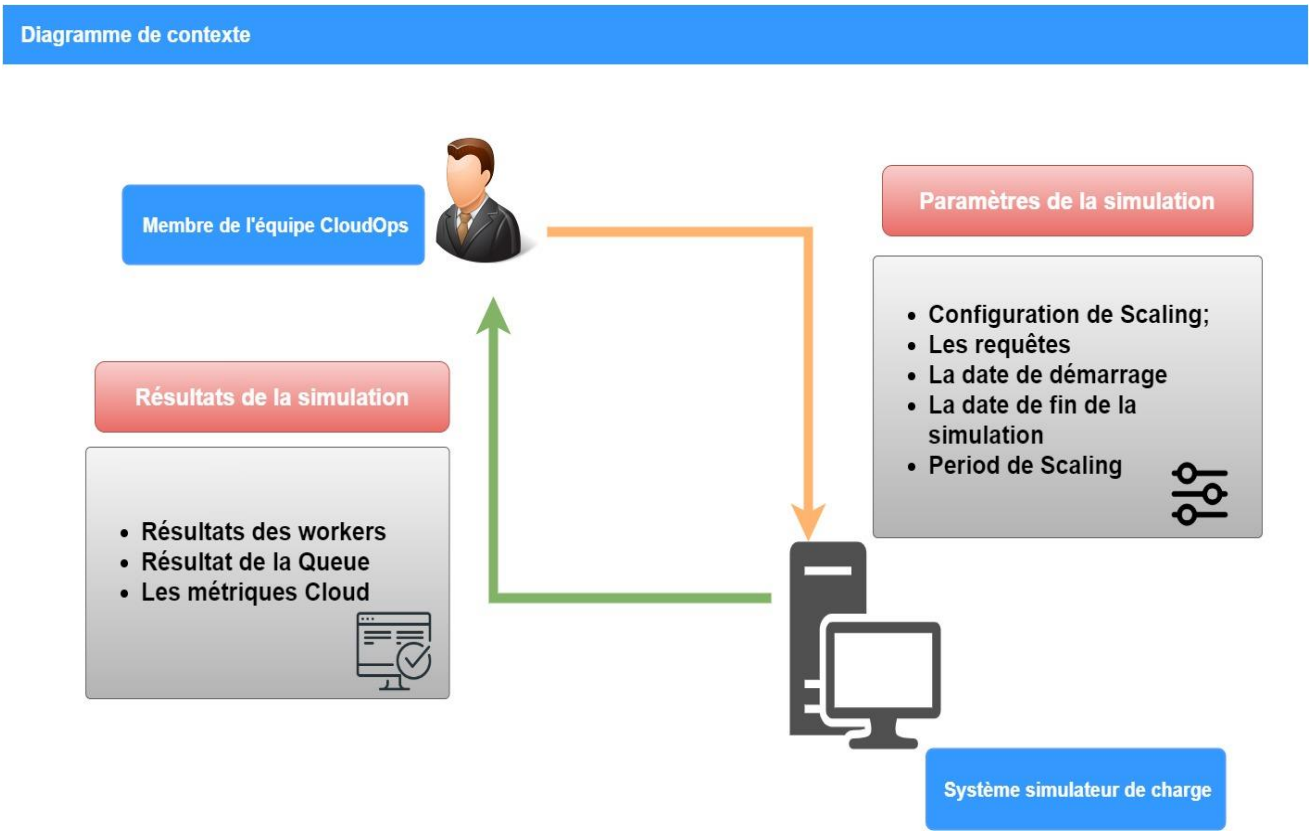

Figure 15: Diagramme de contexte

5. Conclusion

Dans ce chapitre, j'ai procédé à une étude préliminaire sur le service d'Auto Scaling en donnant une idée plus claire sur l'existant et ses limites, sur les besoins fonctionnels et techniques pour la simulation du service ainsi qu'un état de l'art sur les approches de simulation, terminée par une modélisation du contexte de l'outil de simulation.

Dans la partie suivante, j'entamerai les spécifications générales du projet sur le plan fonctionnel et technique.

Chapitre 3 : Spécifications Générales

Dans ce chapitre je présenterai d'une façon détaillée toutes les spécifications fonctionnelles et techniques qui seront intégrés et prises en considération lors de l'implémentation.

1. Spécifications fonctionnelles

1.1 Diagramme de cas d'utilisation globale

L'identification des cas d'utilisations nous donne un aperçu des fonctionnalités futures à implémenter.

Notre système est composé de trois cas d'utilisations globales. Chaque cas d'utilisation est détaillé en sous fonctionnalités.

Nous présentons ci-dessous le diagramme de cas d'utilisations :

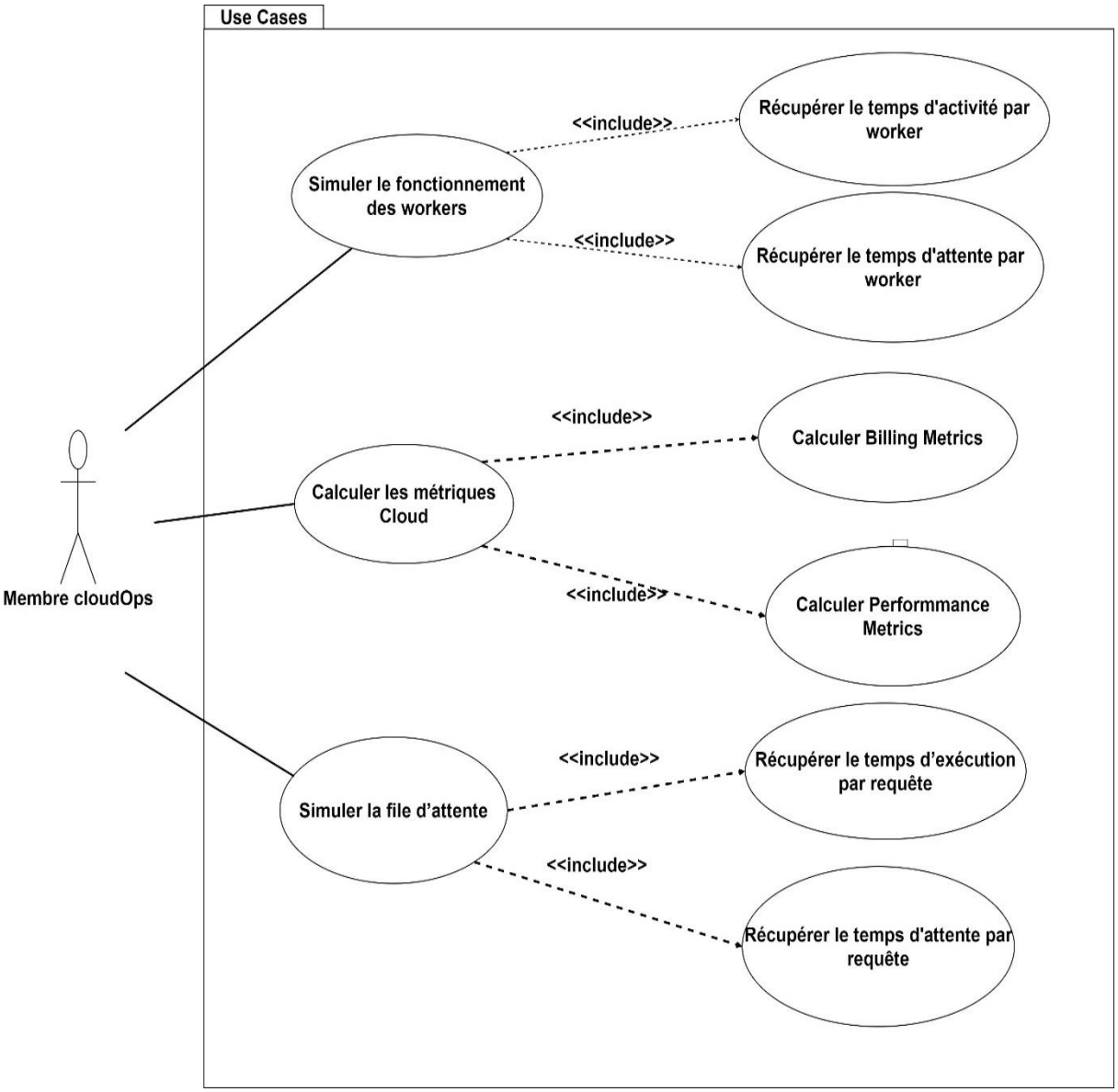

Figure 16: Diagramme de cas d'utilisations

1.2 Description des cas d'utilisations

Cette partie a pour objectif de décrire les fonctionnalités offertes par notre système d'une façon plus détaillée.

a) Simuler les Workers

Nous présentons ci-après, la fiche descriptive pour use Case globale « simuler les workers » :

Titre	Simuler les Workers
Résumé	Afficher les résultats de chaque Worker au cours de la simulation.
Acteur	Membre de l'équipe CloudOps
Sous fonctionnalités	• Calculer le temps d'attente par worker ; • Calculer le temps d'activité ; • Récupérer la date de démarrage par Worker; • Récupérer la date d'éteinte par worker;

Tableau 2: Fiche descriptive UseCase- Simuler les workers

b) Simulateur de la queue

Nous présentons ci-après, la fiche descriptive pour Use Case « simuler la queue » :

Titre	Simuler la file d'attente
Résumé	Calculer le temps d'exécution et temps d'attente de chaque requête.
Acteur	Membre de l'équipe CloudOps
Sous fonctionnalités	• Calculer le temps d'attente dans la queue par requête ; • Calculer le temps d'exécution totale par requête ; • Récupérer ID du Worker qui a exécuté la requête ;

Tableau 3: Fiche descriptive UseCase - Simuler la queue

c) Calculer les métriques Cloud

Nous présentons ci-après, la fiche descriptive pour Use Case « Calculer métrique cloud » :

Titre	Calculer Les métriques Cloud de la simulation
Résumé	Afficher les métriques de performance et les métriques de facturations
Acteur	Membre de l'équipe CloudOps

Sous fonctionnalités	• Calculer le montant dû au fournisseur cloud pour l'allocation des workers ; • Calculer la moyenne d'efficacité des workers • Calculer le temps d'attente total des workers ; • Calculer le montant dû au fournisseur cloud pour l'allocation des workers durant le temps d'attente, • Calculer le temps durant lequel le service était indisponible • Calculer le pourcentage du temps dans lequel les Workers étaient indisponible, • Calculer la moyenne du temps d'attente des requêtes dans la queue ; • Calculer le temps d'attente maximal dans la queue • Calculer Percentage des requêtes qui sont passé à la phase d'exécution sans attendre dans la file ;

Tableau 4: Fiche descriptive UseCase – Calculer les métriques Cloud

2. Spécifications techniques

Durant cette phase je vais présenter l'architecture logicielle adoptée, citer les Design patterns utilisés par la suite.

2.1 Architecture de l'application

Dans cette partie, on va définir les composants nécessaires à la construction de l'architecture technique.

a) Architecture logique

Pour avoir une vue globale sur l'organisation des éléments du système et afin de proposer une structure à la fois simple et performante, une architecture monolithique a été adoptée.

La figure suivante montre l'architecture logique de notre solution :

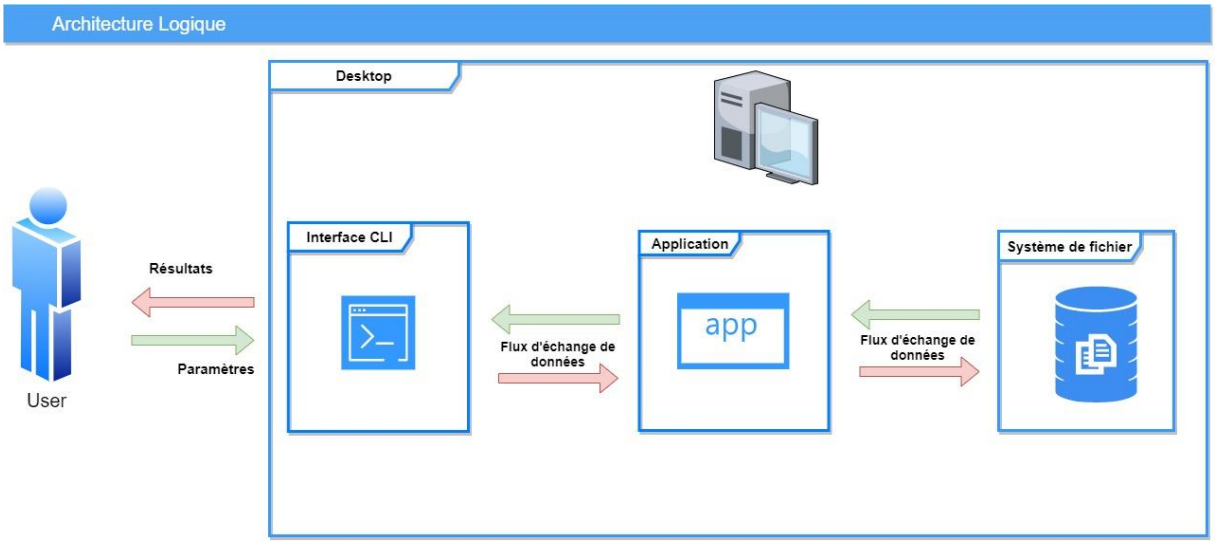

Figure 17: Architecture logique

b) Architecture logicielle

Dans cette partie nous allons décrire d'une manière symbolique et schématique les différents éléments du système, leurs interrelations ainsi que leurs interactions.

Contrairement aux spécifications produites par l'analyse fonctionnelle qui décrivent ce que doit réaliser l'application, le modèle d'architecture logicielle répond à ces spécifications en décrivant le ''comment faire''.

L'architecture en couches est la conséquence inéluctable d'une approche qui s'appuie sur la réalisation de composants réutilisables.

Dans le but de réaliser un système puissant, évolutif et modulaire nous allons adopter une architecture en couches qui nous garantit le maximum de découplage entre les couches logicielles mises en œuvre.

Le schéma ci-dessous montre l'architecture logicielle pour le développement de notre application :

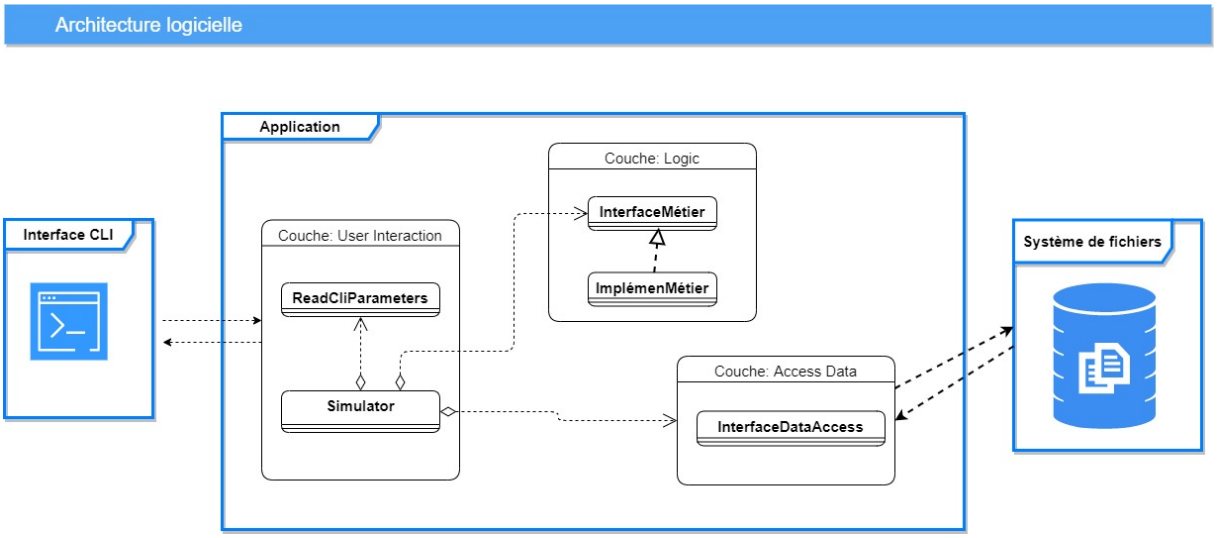

Figure 18: Architecture logicielle

2.2 Design Patterns

Les design patterns se présentent comme étant des solutions de conception communes à un problème récurrent dans un contexte donné. Ainsi, leur usage apporte une évolutivité, une lisibilité et une efficacité au développement. Par ailleurs, ils offrent un transfert de compétence rapide et conception orientée objet, dans la mesure où ils représentent un catalogue des meilleurs pratiques à adopter.

A ce titre nous présentons ci-dessous les designs patterns dont nous avons fait usage dans la conception de notre projet.

- *Inversion de contrôle (IOC)*

L'inversion de contrôle est « un patron d'architecture » visant à réduire le couplage inter ou intra couches applicatives. Ce pattern est aussi connu sous le nom d'injection de dépendance.

Il concerne la manière dont un objet obtient une référence vers ses dépendances. L'approche habituelle pour obtenir ces dépendances au sein d'un objet, que ce soit via un singleton ou un autre mécanisme de Lookup (Service Locator, instanciation directe, ...), a le désavantage d'introduire un couplage fort entre l'objet et ce mécanisme de Lookup.

Avec l'inversion de contrôle, les dépendances de l'objet lui sont passées à travers son constructeur ou par ses setters une fois instancié « d'où le nom du pattern injection des dépendances ».

3. Conclusion

Au cours de chapitre, nous avons énoncé les spécifications fonctionnelles et techniques de ce système au cours desquelles les besoins fonctionnels ont été traduits en diagramme de de cas d'utilisations et les besoins techniques par l'architecture logique et logicielle de l'application.

Le chapitre suivant portera sur l'analyse et la conception du système où je présenterai les diagrammes de classes et les diagrammes de séquences

Chapitre 4 : Analyse et conception

Ce chapitre sera consacré à l'Analyse et la Conception suivant la démarche XP. Plusieurs itérations seront présentées en commençant par l'identification des taches, puis on enchainera par un diagramme de classes candidates, et puis un diagramme de séquence détaillée pour l'itération.

Itération 1 : Récupération des données et implémentation des objets de base

1.1 Identification des tâches de l'itération

- Documentation sur le fonctionnement d'un worker cloud
- Documentation sur l'objet de configuration Json
- Extraction des paramètres de la simulation à partir d'un fichier csv
- Extraction des requêtes clients à partir d'un fichier csv
- Extraire une unité de configuration à partir d'un fichier Json
- Conception des interfaces : Configuration, ClientRequest, et Worker
- Implémentation des interfaces
- Implémenter les tests unitaires

La figure suivante montre la composition de fichier de paramètres de la simulation :

Figure 19: Fichier de paramètres

L'image ci-dessous montre la composition d'une unité de configuration :

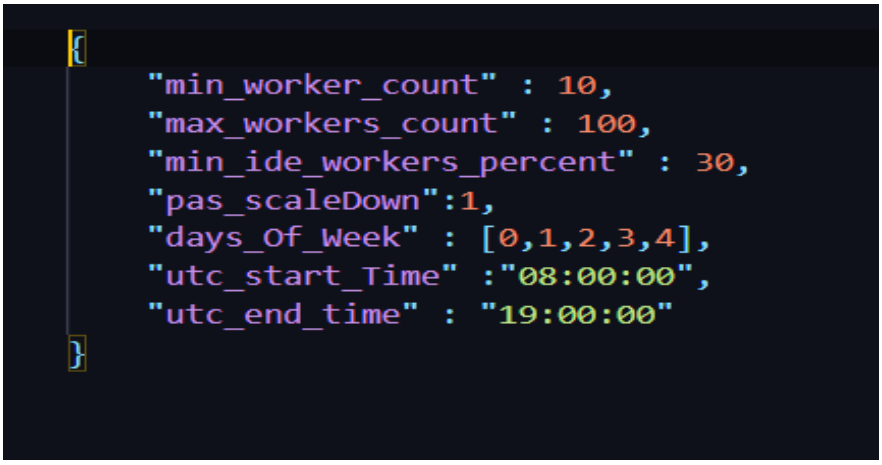

Figure 20: Unité de configuration- itération 1

La figure suivante montre un échantillon de requêtes :

A	B	C
ID	ArrivedTime	ExecutionTime(s)
1	20/01/2020 9:00:00	10
2	20/01/2020 9:00:06	5
3	20/01/2020 9:00:10	2
4	20/01/2020 9:00:11	1
5	20/01/2020 9:00:11	3
6	20/01/2020 9:01:29	3
7	20/01/2020 10:00:00	1
8	20/01/2020 10:00:00	7
9	20/01/2020 10:00:02	8
10	20/01/2020 10:00:30	2

Figure 21 : Exemple de Fichier de requêtes

1.2. Identification des classes candidates

L'analyse de cette itération a permis de lire les fichiers inputs dont la structure est décrite via les figures ci-dessous :

Classe	Description
worker	Désigne le worker cloud
ClientRequest	Désigne la requête d'un client
Configuration	Désigne une unité de configuration
SimulationDateTime	Désigne le temps réel à chaque instant da la simulation
WorkerSatatus	Désigne le statut d'un worker
Parameters	Désigne les paramètres de la ligne de commande
FileReader	Class static qui implémente les fonctionnalités de lecture de fichiers

Tableau 5: Tableau de classes candidates - itération 1

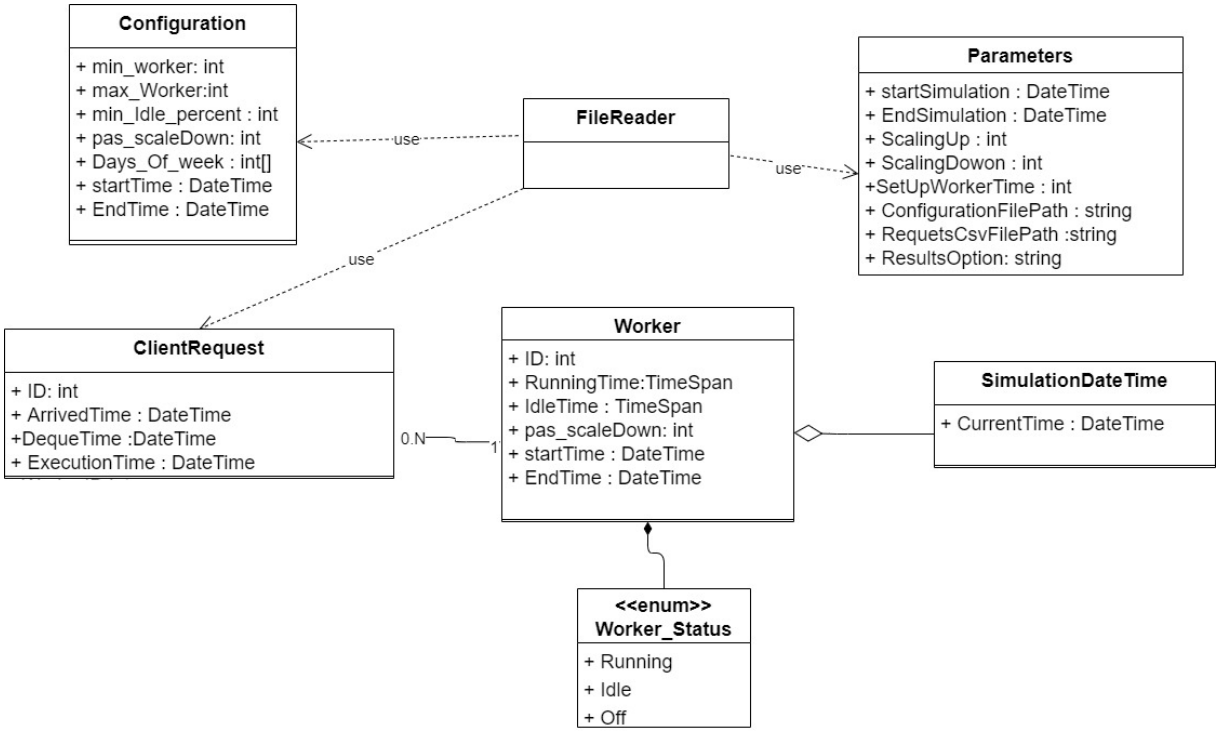

Figure 22: Diagramme de classes candidates - itération 1

1.3 Diagramme de séquences détaillé - l'itération 1

La figure suivante montre le diagramme de séquence de la première itération :

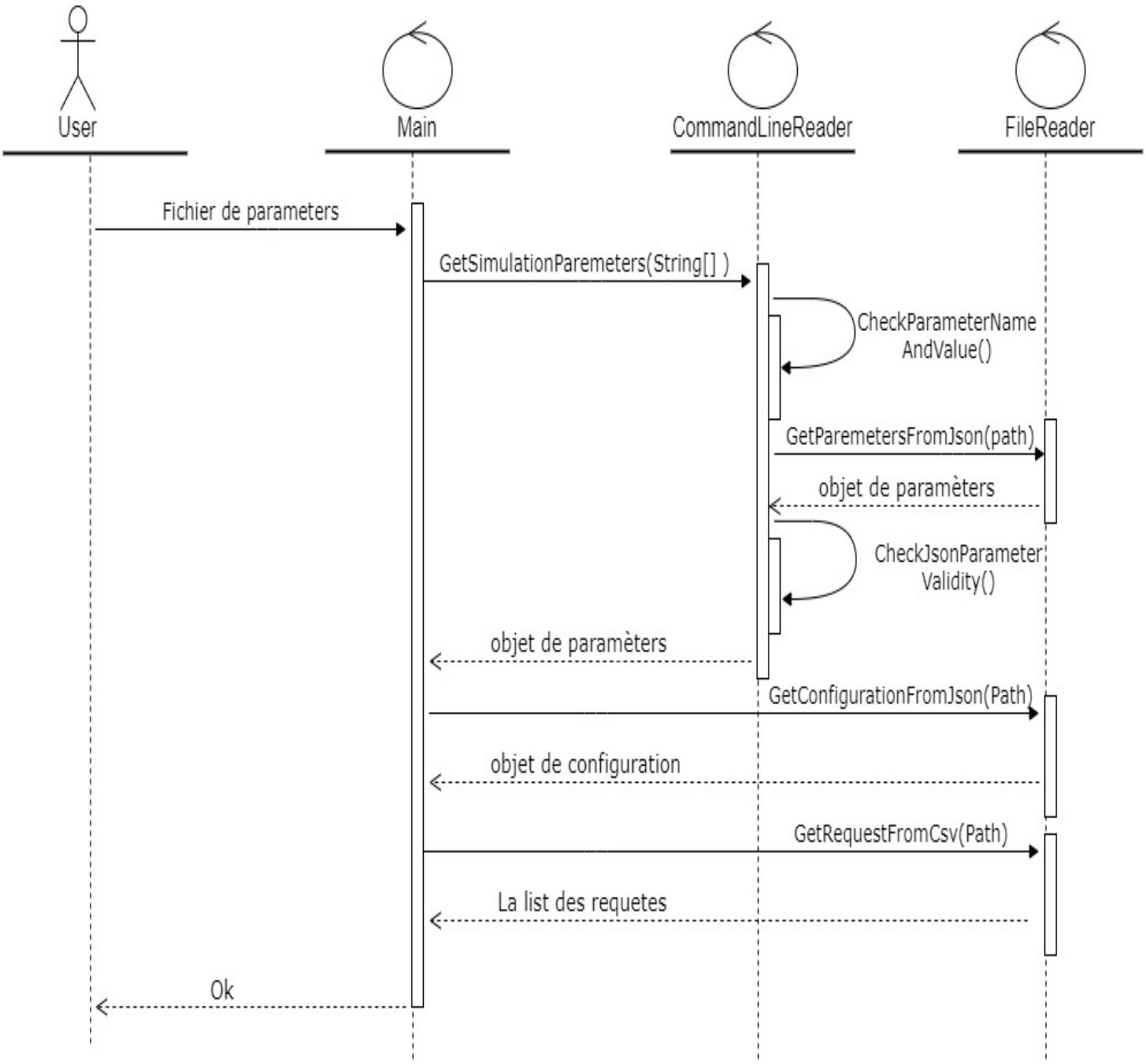

Figure 23: Diagramme de séquences détaillé - itération 1

Itération 2 : Implémentation du QueueSimulater et WorkersSimulater

2.1 Identification des tâches de l'itération

- Analyse des événements de la queue (enqueue, GetRequest,...)

- Analyse des fonctionnalités de SimulatorWorkers (addWorker, shutDownWorker, DequeueRequest…)
- Conception de l'interface SimulaterQueue & SimulaterWorkers
- Implémenter l'interface SimulaterCloud & SimulaterWorkers
- Implémenter les tests unitaires

2.2. Identification des classes candidates

Classe	Description
workersSimulater	Désigne le simulateur de la liste des workers
Worker	Désigne le worker cloud
QueueSimulater	Désigne la liste des requêtes
ClientRequest	Désigne la requête d'un client
SimulationDateTime	Désigne le temps réel à chaque instant da la simulation

Tableau 6: Tableau de classes candidates - itération 2

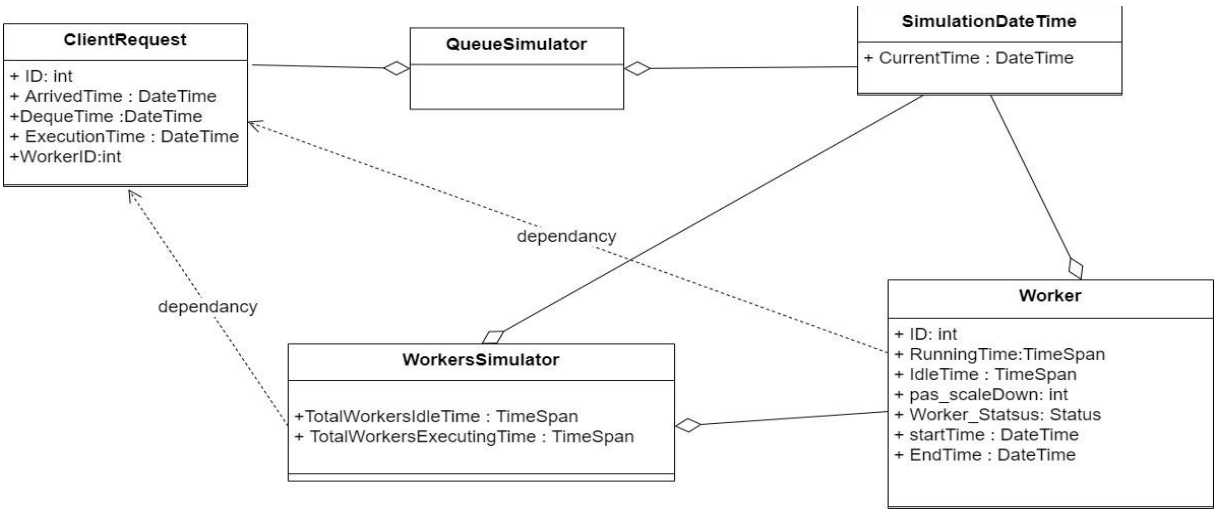

Figure 24: Diagramme de classes candidates - itération 2

Itération 3 : Implémenter le ScalerSimulater

3.1 Identification des tâches de l'itération

- Analyse des fonctionnalités de base de service de scaling (ScaleUp, ScaleDown, GetNextScalingDate ...)
- Conception de l'objet ScalerSimulater
- Implémenter l'interface ScalerSimulater

- Implémenter les tests unitaires

3.2. Identification des classes candidates

Classe	Description
ScalerSimulater	Désigne le simulateur de service de scaling
workersSimulater	Désigne le simulateur de la liste des workers
SimulationDateTime	Désigne le temps réel à chaque instant da la simulation
Worker	Désigne le worker cloud
ClientRequest	Désigne la requête d'un client
Configuration	Désigne une unité de configuration

Tableau 7: Tableau de classes candidate - itération 3

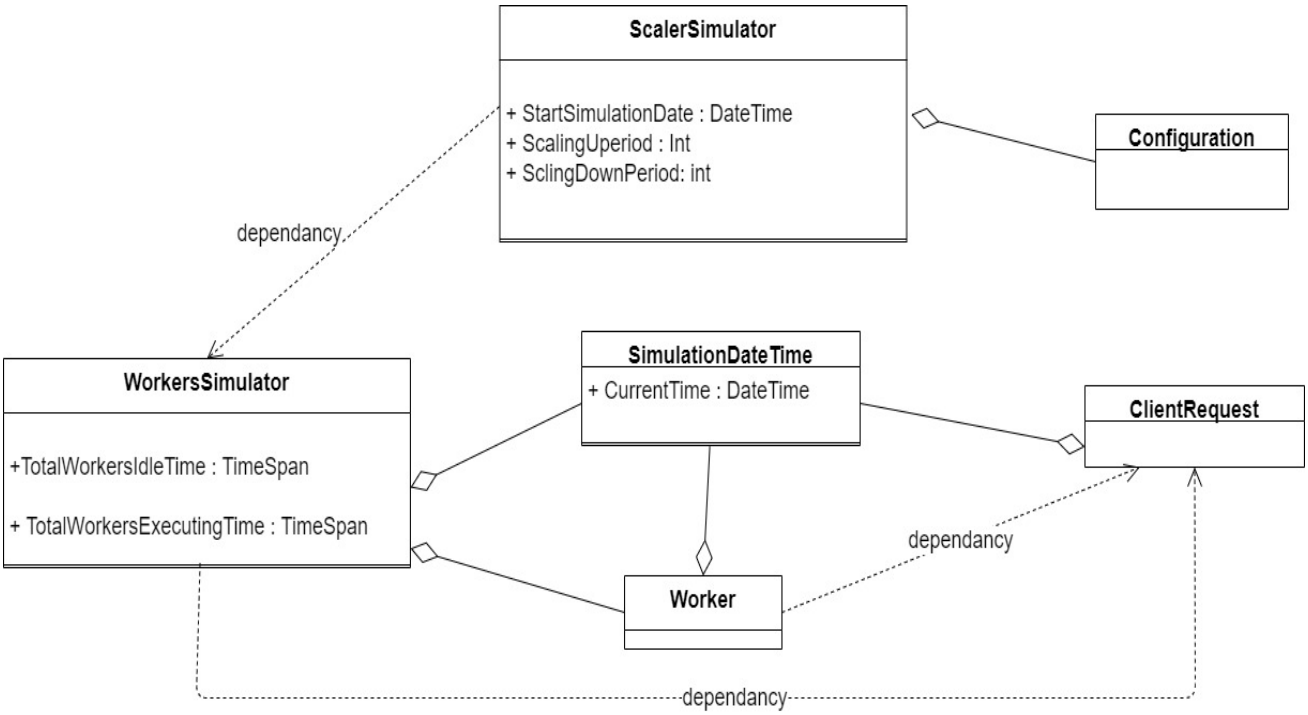

Figure 25: Diagramme de classes candidates - itération 3

Itération 4 : Implémenter Simulator & Orchestrator

4.1 Identification des tâches de l'itération

- Conception de l'objet Orchestrator (GetNextEvent...)
- Conception de l'objet simulateur (StartSimulation,EndSimulation)
- Implémenter l'objet DateTimeSimulation
- Gérer les dépendances du temps entre les différentes interfaces (Injecter la dépendance DateTimeSimulation)
- Les tests unitaires
- Les tests d'intégration

4.2. Identification des classes candidates

Classe	Description
Orchestrator	L'objet d'orchestration entre les évents de la simulation
Simulator	C'est le simulateur qui prend et traite des évents
SimulationEvent	Désigne le prochain évent qui se produira
WorkersSimulator	Désigne le simulateur de la liste des workers
QueueSimulator	Désigne la liste des requêtes
ScalerSimulator	Désigne le simulateur de service de scaling

Tableau 8: Tableau de classes candidates - itération 4

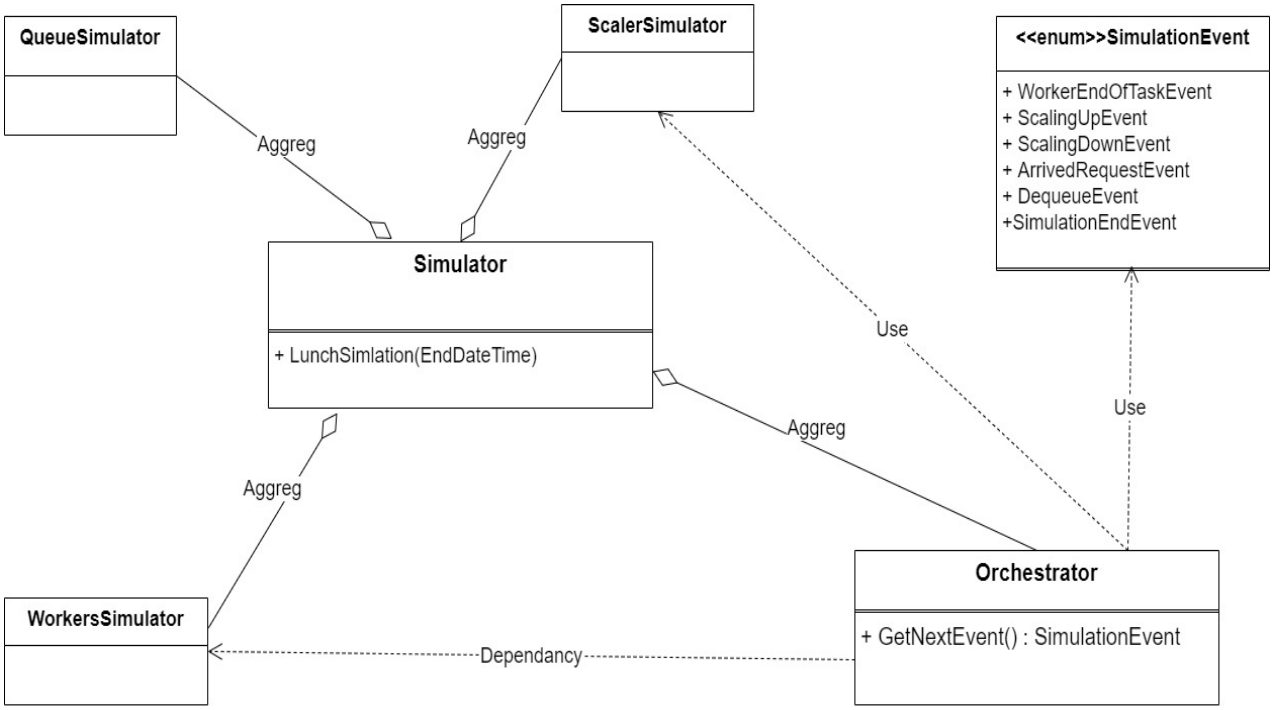

Figure 26: Diagramme de classes candidates - itération 4

4.3 Diagramme de séquences détaillé - l'itération 4

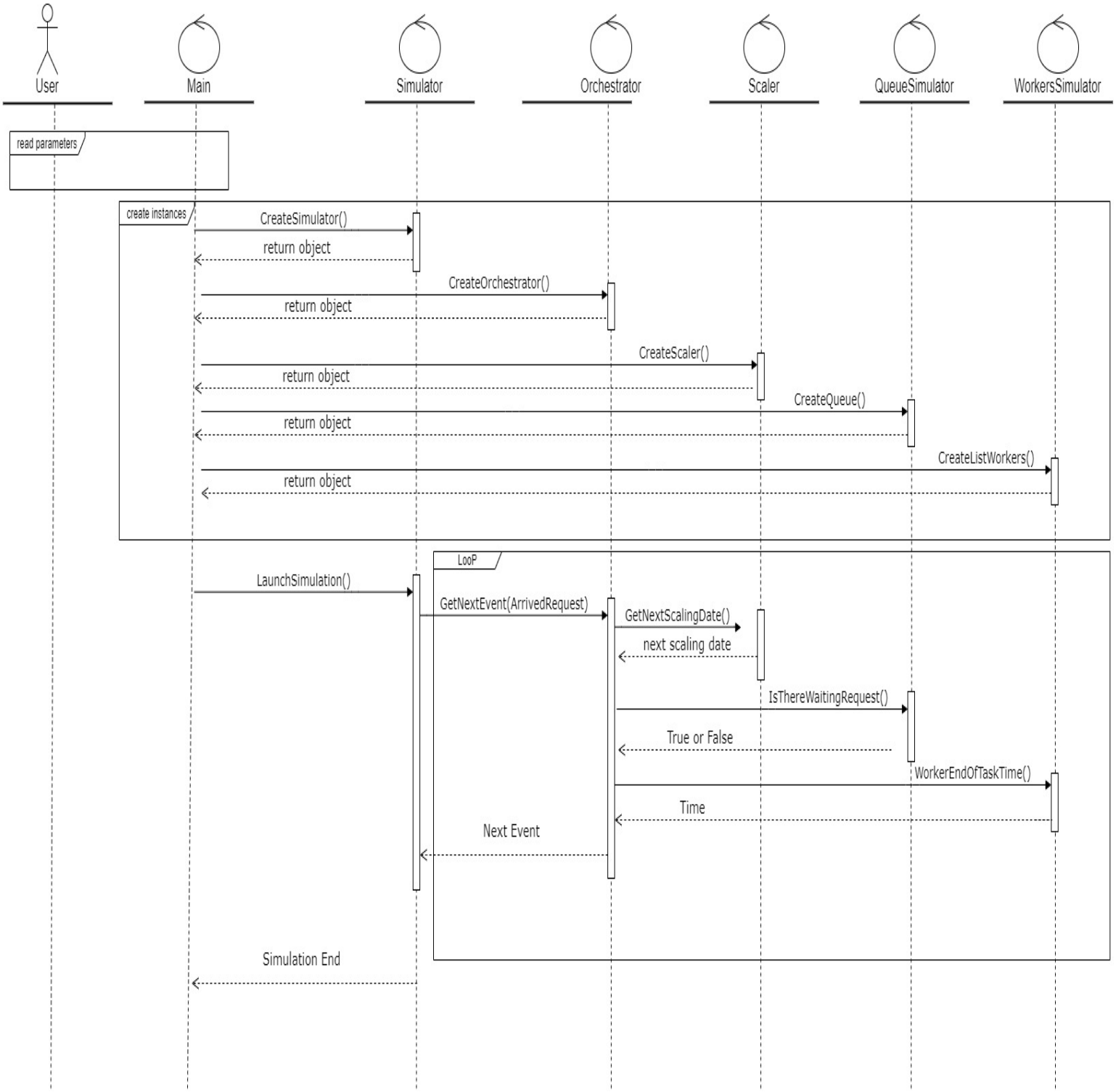

Figure 27: Diagramme de séquences détaillé - itération 4

Itération 5 : extension de l'objet de configuration « IConfigurationHolder »

5.1 Identification des tâches de l'itération

- Extension de l'objet configuration pour couvrir les jours de la semaine
- Conception de l'interface IConfigurationHolder qui représente une configuration hebdomadaire
- Implémenter l'interface
- Intégrer IConfigurationHlder à la solution existante
- Implémenter les tests unitaires
- Implémenter les tests d'intégration

La figure suivante montre la composition de fichier configuration dans cette itération :


```
{
    "Pull Client 1":[
    {
        "min_worker_count" : 50,
        "max_workers_count" : 300,
        "min_ide_workers_percent" : 30,
        "pas_scaleDown":5,
        "days_Of_Week" : [0,1,2,3,4],
        "utc_start_Time" :"08:00:00",
        "utc_end_time" : "19:00:00"
    },
    {
        "min_worker_count" : 10,
        "max_workers_count" : 100,
        "min_ide_workers_percent" : 30,
        "pas_scaleDown":10,
        "days_Of_Week" : [5],
        "utc_start_Time" :"09:00:00",
        "utc_end_time" : null
    },
    {
        "min_worker_count" : 1,
        "max_workers_count" : 10,
        "min_ide_workers_percent" : 30,
        "pas_scaleDown": 3,
        "days_Of_Week" : [6],
        "utc_start_Time" : null,
        "utc_end_time" : "20:00:00"
    }
    {
        "min_worker_count" : 20,
        "max_workers_count" : 200,
        "min_ide_workers_percent" : 30,
        "pas_scaleDown":5,
        "days_Of_Week" : null,
        "utc_start_Time" : null,
        "utc_end_time" : null
    }
]}
```


Figure 28: Fichier de configuration - itération 5

5.2. Identification des classes candidates

Classe	Description
ConfiguartionHolder	Désigne configuration d'un pull
Configuration	Désigne une unité de configuration
FileReader	Static class pour la lecture des fichiers
ScalerSimulater	Désigne le simulateur de service de scaling

Tableau 9: Tableau de classes candidates - itération 5

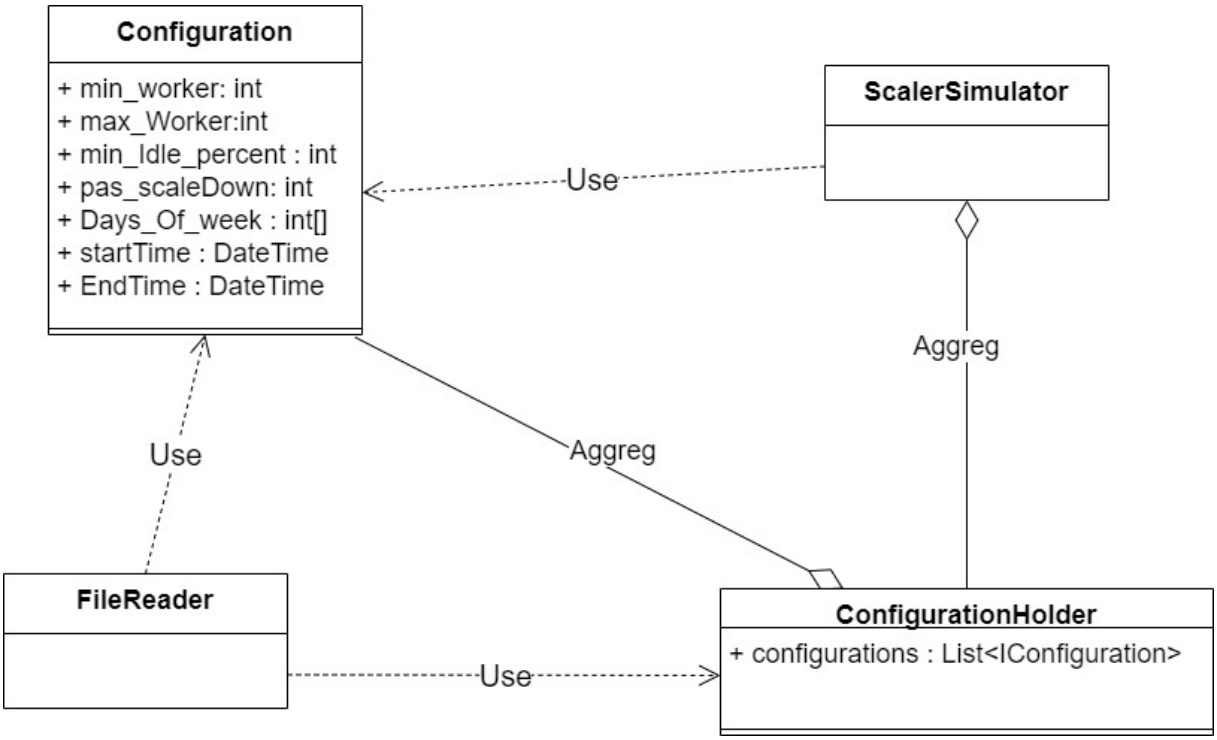

Figure 29: Diagramme de classes candidates - itération 5

Itération 6 : Implémenter les Métriques

6.1 Identification des tâches de l'itération

- Benchmarking Métriques cloud les plus expressives
- Documentation sur la tarification de Azure Cloud
- Conception de l'interface MetricCloud (getIdleTime,GetCost...)

- Implémentation de l'interface MetricCloud
- Génération de fichier pour les metrics Cloud : metrics.csv
- Les tests unitaires
- Les tests d'intégration

6.2 Identification des classes candidates

Classe	Description
SimulationMetrics	Les métriques Cloud de la simulation
WorkersSimulator	Désigne le simulateur de la liste des workers
QueueSimulator	Désigne la liste des requêtes
SimulationDateTime	Désigne le temps réel à chaque instant da la simulation

Tableau 10: Tableau de classes candidates - itération 6

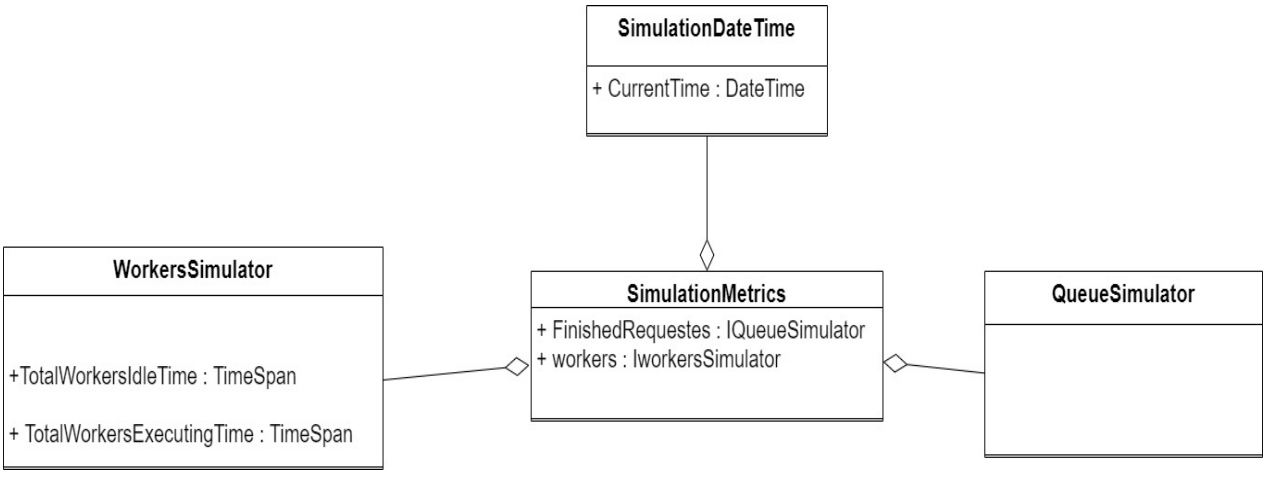

Figure 30: Diagramme de classes candidates - itération 6

Itération 7 : Implémentation du Simulation ResultPrinter

7.1 Identification des tâches de l'itération

- Conception de l'interface SimulationResultPrinter
- Implémentation de SimulationResultPrinter
- Générer le fichier de résultats des workers

- Générer un fichier de résultas des requestes
- Générer un fichier de résultats des requestes
- Afficher les résultats de la simulation

7.2 Identification des classes candidates

Classe	Description
SimulationMetrics	Les métriques Cloud de la simulation
WorkersSimulator	Désigne le simulateur de la liste des workers
QueueSimulator	Désigne la liste des requêtes
SimulationDateTime	Désigne le temps réel à chaque instant da la simulation
Parameters	Désigne l'objet de paramètres de la simulation

Tableau 11: Tableau de classes candidates - itération 7

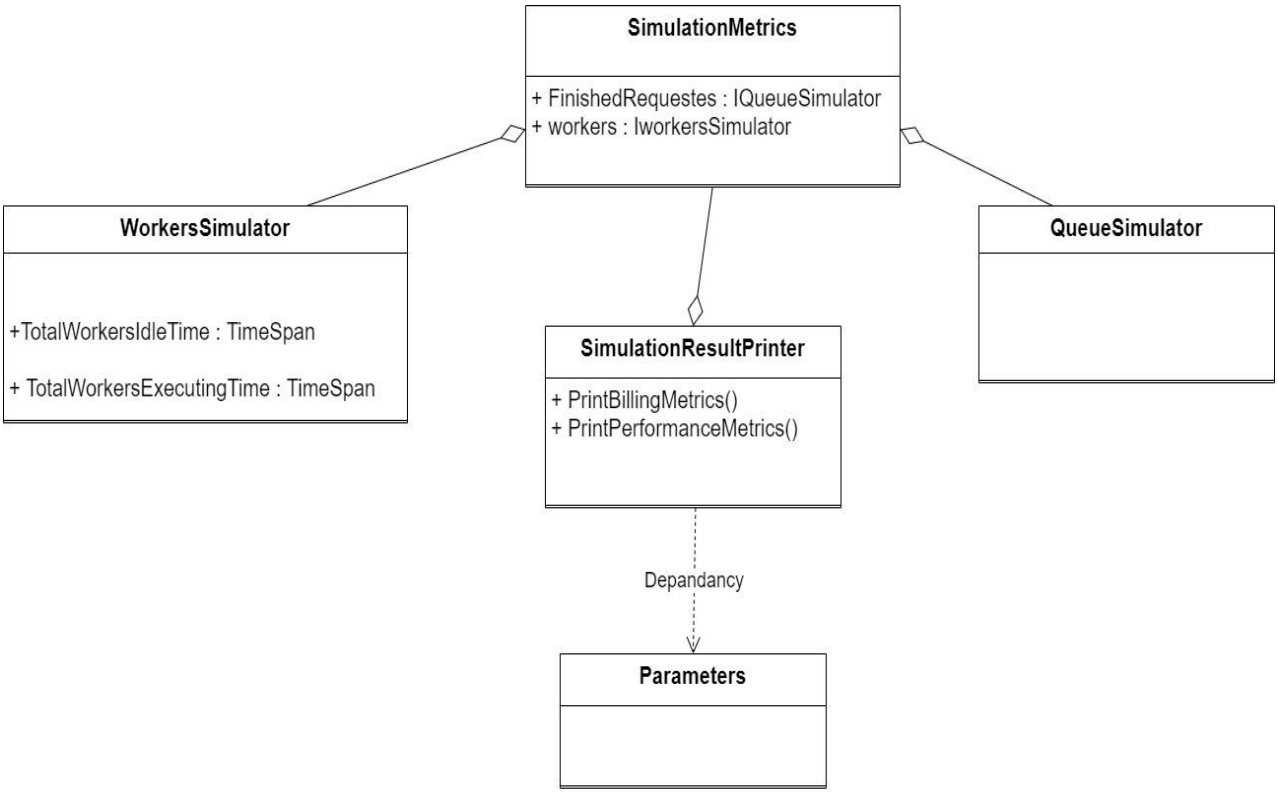

Figure 31: Diagramme de classes candidates - itération 7

7.3 Diagramme de séquences détaillé - l'itération 7

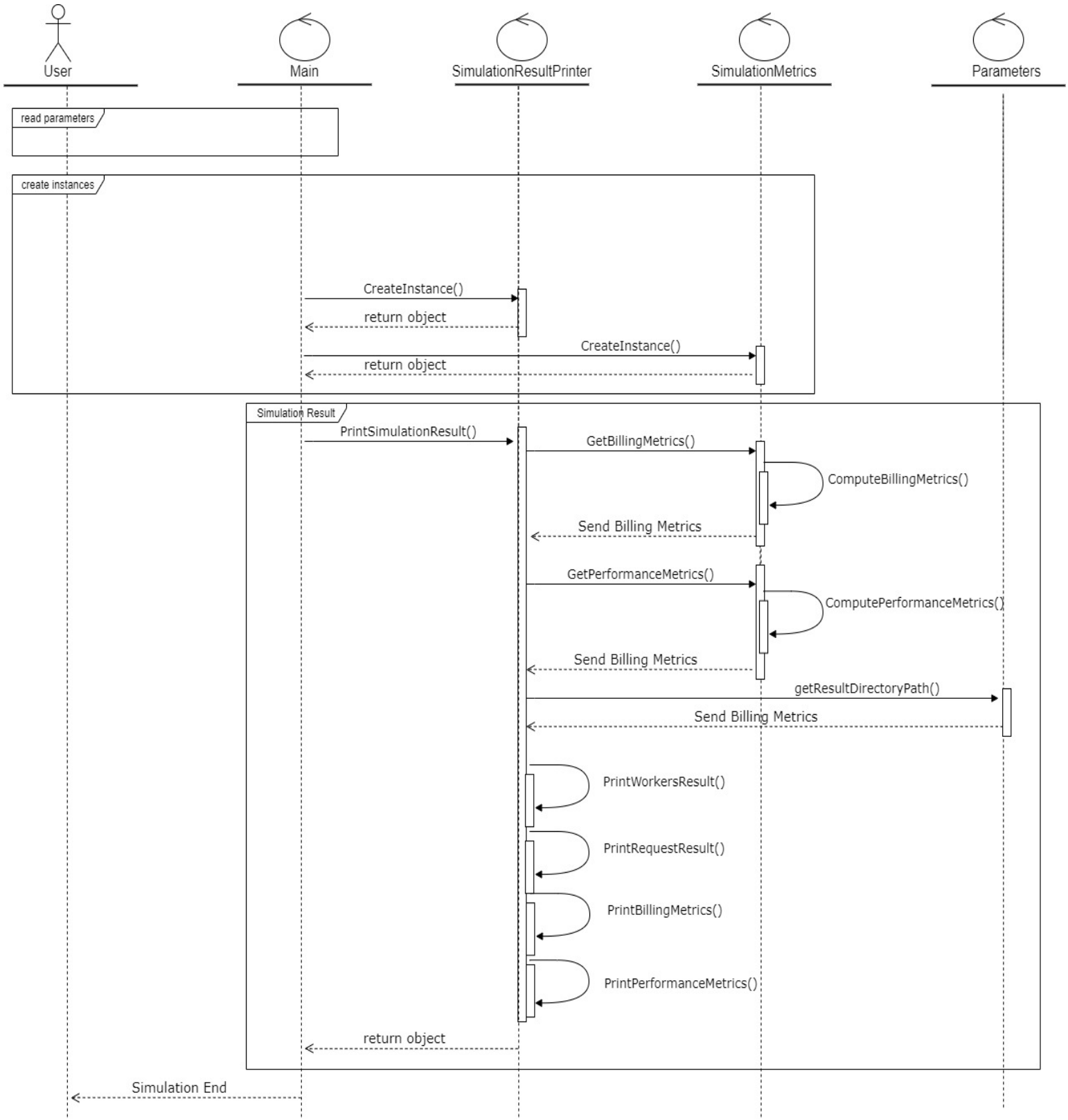

Figure 32: Diagramme de séquences - itération 7

8. Conception générale de l'application

La conception générale du système projette la partie fonctionnelle sur l'architecture logicielle selon les couches qu'elle présente à savoir : la couche UserInteraction, la couche Logic, la couche AccessData. Nous présentons dans un premier temps un schéma globale illustrant cette projection dans la figure suivante.

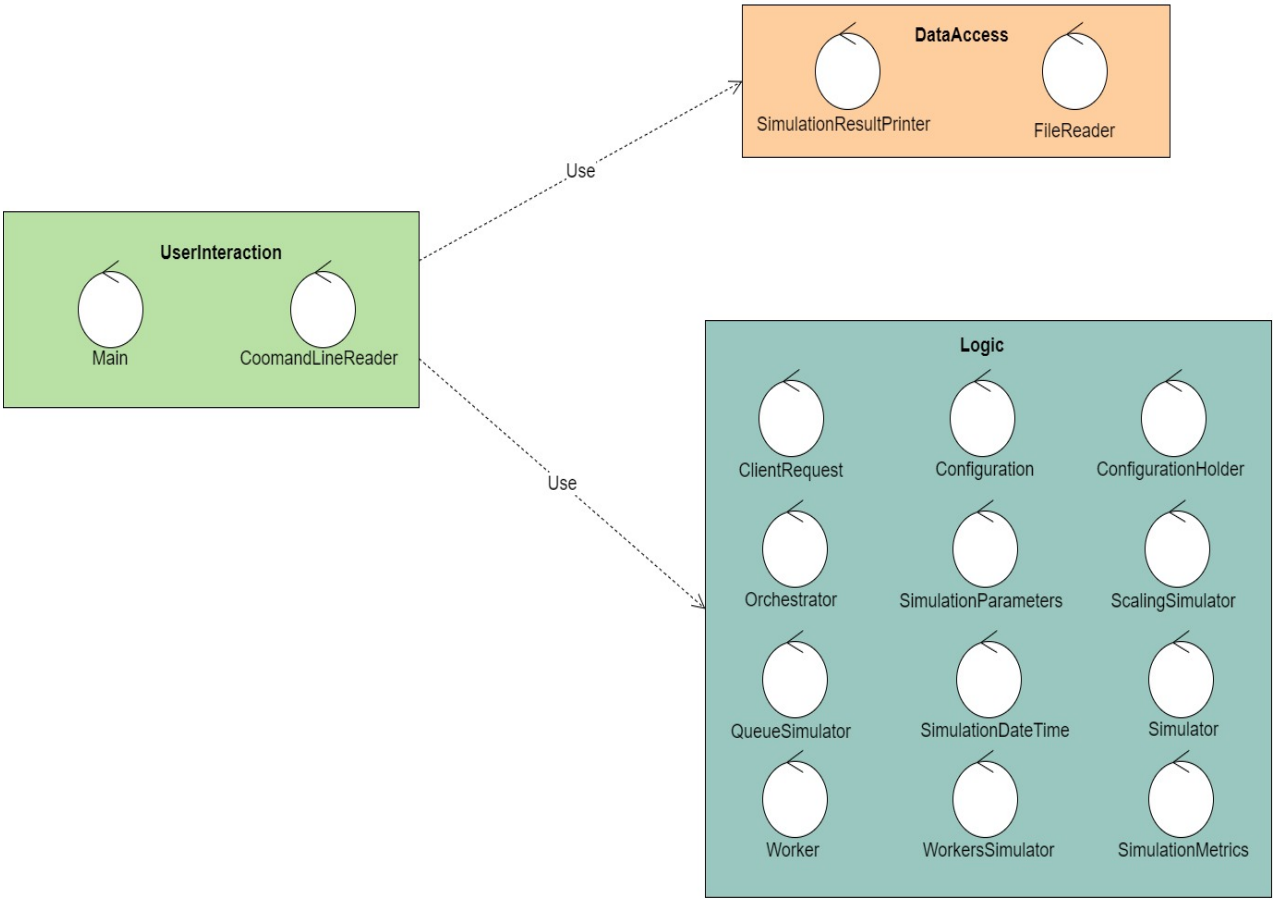

Figure 33: Conception générale du système

9. Conclusion

Dans ce chapitre, j'ai présenté une étude analytique des différentes itérations, d'une vue statique et d'une autre dynamique.

Dans le chapitre suivant, il s'agira de transformer ces diagrammes en système fonctionnel.

Chapitre 5 : Réalisation

Ce chapitre qui clôture mon rapport est dédié à la réalisation et la mise en oeuvre de l'application. Dans cette partie, je vais commencer par parcourir en détail l'environnement technique du projet. J'étalerai par la suite les différentes étapes de mise en oeuvre de l'application.

1. Outils et technologies utilisés

1.1 Environnement de développement

Je présenterai dans cette partie les outils et les technologies utilisés dans le développement de l'application.

a) .NET Framework

Figure 19 : .Net Framwork

D'après la documentation de Micrososft .Net Framwork, ce dernier est un sous ensemble de la technologie NET. C'est une Infrastructure de développement s'appuyant sur la norme Common Language Infrastructure. DotNET Framework est utilisé pour développer la partie extraction de données ainsi que pour le déploiement de l'outil finale car c'est la technologie utilisée au sein de la société ainsi que plusieurs librairies telles que le wrapper des commandes Git et le wrapper des appels http du TeamCity ainsi que la librairie Roslyn utilisé pour le calcul du changement du code utilisé pour l'extraction de données sont développées en .NET par l'équipe INFRA au sein de laquelle s'est développé ce projet.

b) Microsoft Visual Studio 2015

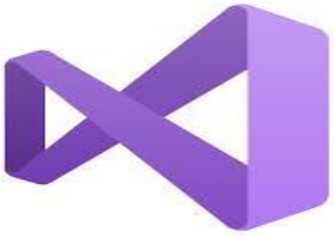

Figure 20 : Visual studio

Visual Studio est une suite de logiciels de développement pour Windows et mac OS conçue par Microsoft.

Visual Studio est un ensemble complet d'outils de développement permettant de générer des applications web ASP.NET, des services web XML, des applications bureautiques et des applications mobiles. Visual Basic, Visual C++, Visual C# utilisent tous le même environnement de développement intégré (IDE), qui leur permet de partager des outils et facilite la création de solutions faisant appel à plusieurs langages. Par ailleurs, ces langages permettent de mieux tirer parti des fonctionnalités du framework .NET, qui fournit un accès à des technologies clés simplifiant le développement d'applications web ASP et de services web Json grâce à Visual Web Developer.

c) C sharp

Figure 34: Logo C sharp

C# est un langage de programmation orientée objet, fortement typé, dérivé de C et de C++, ressemblant au langage Java. Il est utilisé pour développer des applications web, ainsi que des applications de bureau, des services web, des commandes, des widgets ou des bibliothèques de classes. En C#, une application est un lot de classes où une des classes comporte une méthode Main, comme cela se fait en Java3.

C# est destiné à développer sur la plateforme .NET, une pile technologique créée par Microsoft pour succéder à COM.

Les exécutables en C# sont subdivisés en assemblies, en namespaces, en classes et en membres de classe. Un assembly est la forme compilée, qui peut être un programme (un exécutable) ou une bibliothèque de classes (une dll). Un assembly contient le code exécutable en MSIL, ainsi que les symboles. Le code MSIL est traduit en langage machine au moment de l'exécution par la fonction just-in-time de la plateforme .NET.

d) ReSharper 2018

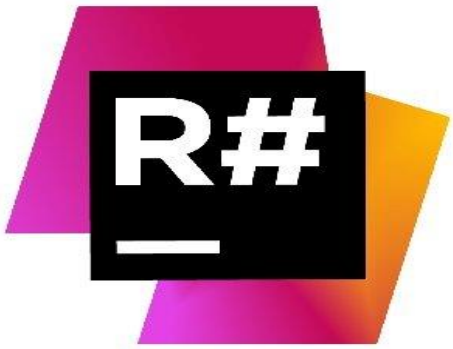

Figure 35 : Logo ReSharper

R# ou Resharper est une extension pour Visual Studio conçu par JetBrains, L'objectif principal de cet outil est d'augmenter la productivité des développeurs de logiciels. Parmi ces fonctions :

- ✓ Analyse statique.
- ✓ Navigation et recherche.
- ✓ Refactoring de code.
- ✓ Amélioration de la complétion automatique.
- ✓ Soutien des tests unitaires.
- ✓ Outils d'architecture (graphes, ...).

1.2 Outil de versioning

a) Git

Figure 36 : Logo Git

Git est un logiciel de gestion de versions décentralisé. C'est un logiciel libre créé par Linus Torvalds, auteur du noyau Linux, et distribué selon les termes de la licence publique générale GNU version 2. En 2016, il s'agit du logiciel de gestion de versions le plus populaire qui est utilisé par plus de douze millions de personnes.

b) Github

Figure 37: Logo Github

GitHub est un service web d'hébergement et de gestion de développement de logiciels, utilisant le logiciel de gestion de versions Git. GitHub propose des comptes professionnels payants, ainsi que des comptes gratuits pour les projets de logiciels libres. Le site assure également un contrôle d'accès et des fonctionnalités destinées à la collaboration comme le suivi des bugs, les demandes de fonctionnalités, la gestion de tâches et un wiki pour chaque projet.

1.3 Les tests

Figure 38: Logo NUnit

NUnit est un framework de tests unitaires open source pour .NET Framework et l'un des nombreux programmes de la famille xUnit. Ces fonctionalites son nombreaux :

- Les tests peuvent être exécutés à partir d'un exécuteur de console, dans Visual Studio via un adaptateur de test, ou via des exécuteurs tiers.
- Les tests peuvent être exécutés en parallèle.
- Prise en charge solide des tests basés sur les données.

- Prend en charge plusieurs plates-formes, notamment .NET Core, Xamarin Mobile, Compact Framework ;
- Chaque scénario de test peut être ajouté à une ou plusieurs catégories, pour permettre un fonctionnement sélectif.

2. Réalisation et mise en œuvre

2.1 Structure du projet

Afin de respecter fidèlement l'architecture logicielle citée précédemment il fallait penser à une structure par couche. Les figures ci-dessous représentent les divers dossiers/packages de mon application.

La figure suivante montre la structure de la solution, qui est composée de deux projets, à savoirs le projet de simulateur et le projet de test unitaire :

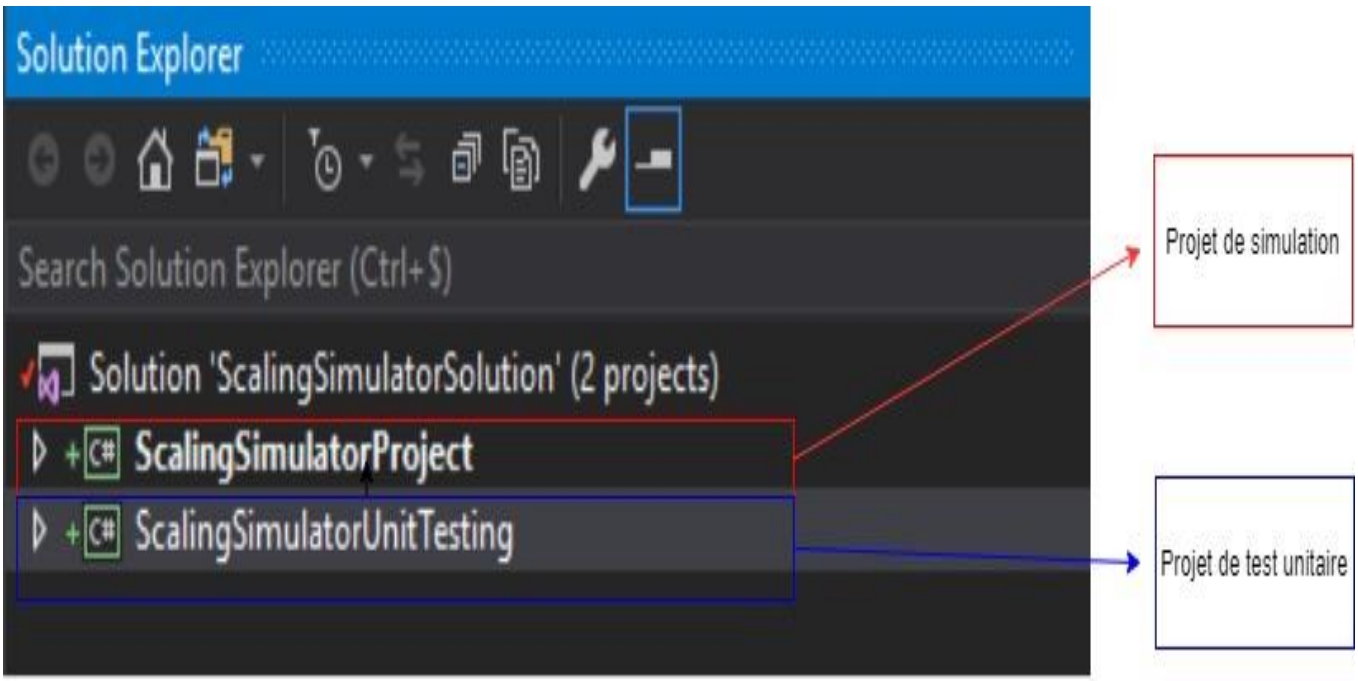

Figure 39: Structure de la solution

La figure suivante montre les différentes packages et couches de notre solution :

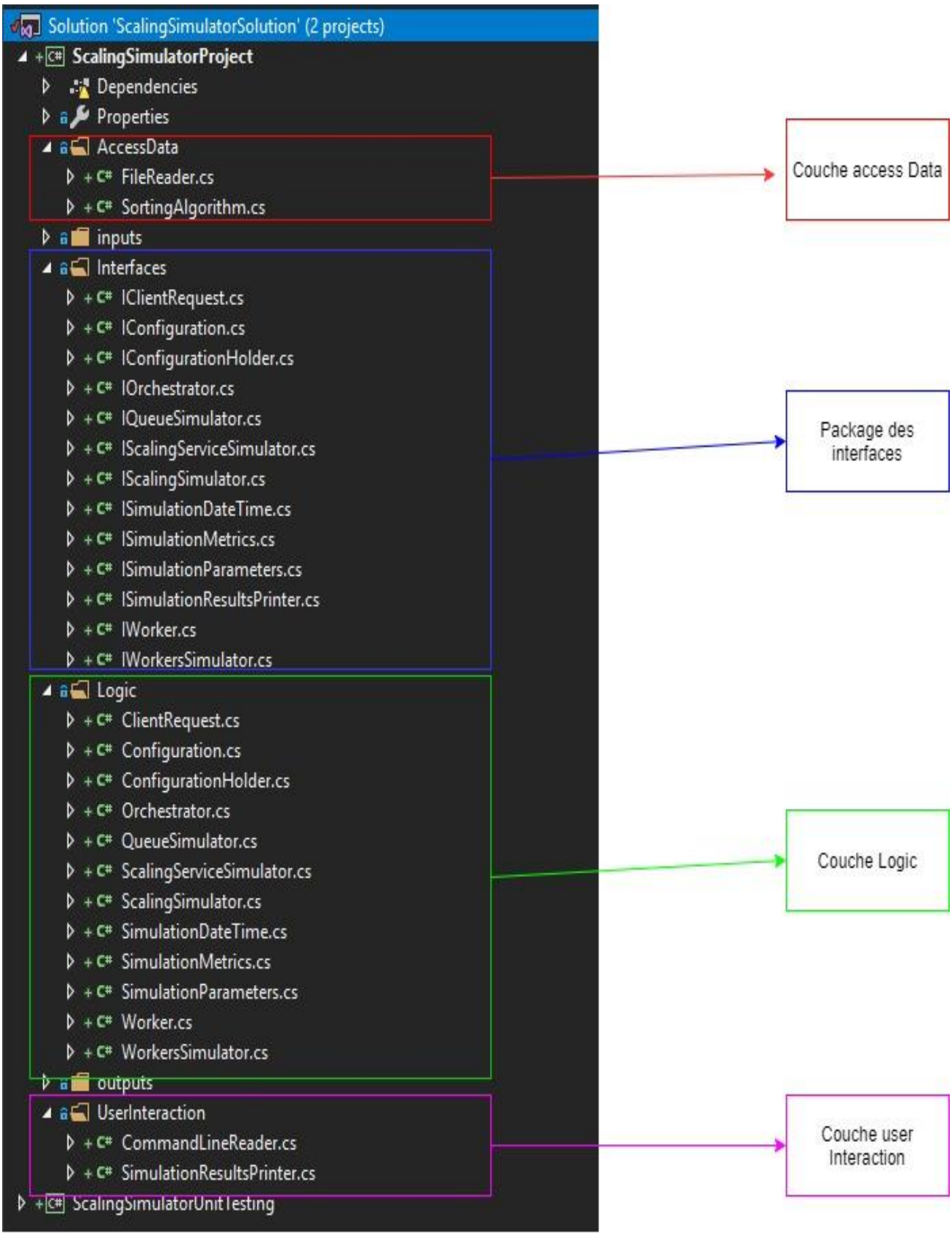

Figure 40: Structure de projet

La figure suivante montre les différentes classes de tests unitaire utilisées au niveau du projet :

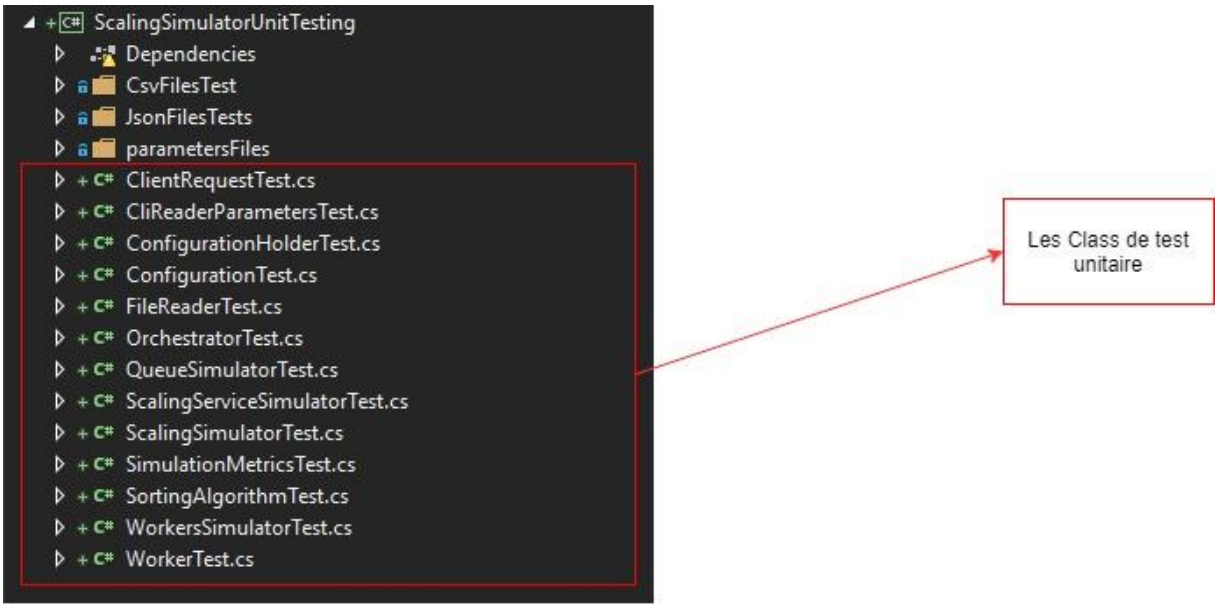

Figure 41: Aperçu sur le contenu du projet de test

2.2 Présentation de l'outil

Dans cette partie je présenterai les inputs, les outputs, ainsi que l'interface de simulateur :

a) Input :

Je présente dans cette partie les différents inputs de simulateur, à savoir le fichier de paramètres, fichier requêtes, ainsi que fichier configuration.

La figure suivante montre la composition du fichier de paramètres :

```
{
    "startSimulationDate":"20/01/2020 00:00:00",
    "endSimulationDate":"27/01/2020 23:59:59",
    "Scaling_up":20,
    "Scaling_down":120,
    "SetUp_workers_Time":30,
    "config_file_Path":"/input/Path/config.json",
    "requests_csv_File" : "/input/Path/requests.csv",
    "workers_State_Output":"/output/path/workers_state.csv",
    "requests_State_Output":  "/output/path/requests_state.csv",
    "Metrics_output":"/output/path/metrics.csv"
}
```

Figure 42: Les inputs de la simulation - Fichier de paramètres

La figure suivante montre le fichier de configuration de service de scaling. Ce fichier est composé des unités de configurations applicables dans différents jours :


```
[
    {
        "min_worker_count" : 50,
        "max_workers_count" : 300,
        "min_ide_workers_percent" : 30,
        "pas_scaleDown":5,
        "days_Of_Week" : [0,1,2,3,4],
        "utc_start_Time" :"08:00:00",
        "utc_end_time" : "19:00:00"
    },
    {
        "min_worker_count" : 10,
        "max_workers_count" : 100,
        "min_ide_workers_percent" : 30,
        "pas_scaleDown":10,
        "days_Of_Week" : [5],
        "utc_start_Time" :"09:00:00",
        "utc_end_time" : null
    },
    {
        "min_worker_count" : 1,
        "max_workers_count" : 10,
        "min_ide_workers_percent" : 30,
        "pas_scaleDown": 3,
        "days_Of_Week" : [6],
        "utc_start_Time" : null,
        "utc_end_time" : "20:00:00"
    },
    {
        "min_worker_count" : 20,
        "max_workers_count" : 200,
        "min_ide_workers_percent" : 30,
        "pas_scaleDown":5,
        "days_Of_Week" : null,
        "utc_start_Time" : null,
        "utc_end_time" : null
    }
]
```


Figure 43: Les inputs de la simulation - Fichier de configuration

La figure suivante montre un échantillon de fichier de requêtes non triées (la tache de tri est incluse au niveau de traitement) :

ArrivedTime	ExecutionTime(ms)
10/06/2020 15:13:01.985	24129001
10/06/2020 15:12:58.040	216325176
10/06/2020 15:13:49.674	23916035
10/06/2020 15:13:45.434	213343
10/06/2020 15:12:43.888	33244
10/06/2020 15:13:04.203	343432
10/06/2020 15:13:57.356	23434
10/06/2020 15:12:41.181	3432
10/06/2020 15:12:59.477	5563
10/06/2020 15:13:05.986	2344
10/06/2020 15:13:52.268	324323
10/06/2020 15:13:58.474	7122
10/06/2020 15:14:28.615	51112
10/06/2020 15:13:27.248	12232
10/06/2020 15:13:31.836	24242
10/06/2020 15:14:14.190	42115
10/06/2020 15:13:39.547	16453
10/06/2020 15:14:25.770	314211
10/06/2020 15:14:26.252	1142
10/06/2020 15:13:55.744	243343
10/06/2020 15:14:30.141	132324
10/06/2020 15:14:53.696	13422
10/06/2020 15:14:26.128	123432
10/06/2020 15:14:05.727	222342
10/06/2020 15:11:01.120	2043423
10/06/2020 15:13:58.018	211113
10/06/2020 15:14:26.158	31113
10/06/2020 15:15:00.511	12432
10/06/2020 15:14:54.851	32133

Figure 44: Les inputs de la simulation - Fichier de requêtes

b) La ligne de commande de simulateur :

La figure ci-dessous illustre la ligne de commande utilisée pour lancer la simulation :

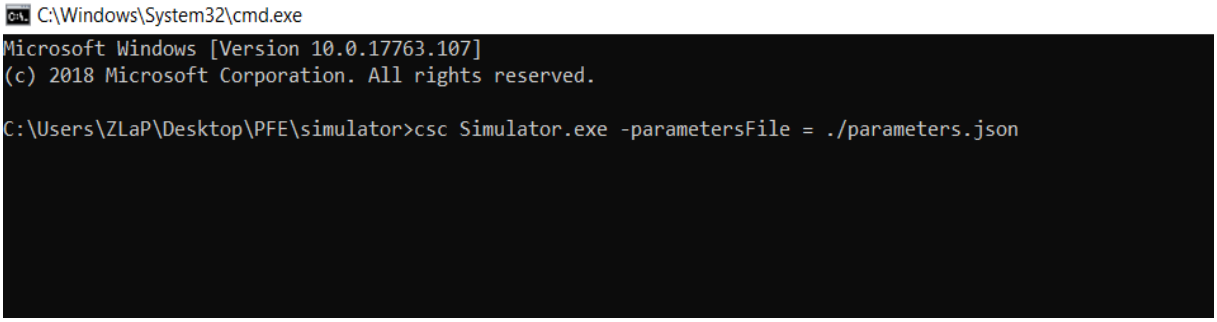

Figure 45: : La ligne de commande de simulateur

c) Outputs :

Je présente dans cette partie les différents outputs de simulateur, à savoir l'état des Workers, l'état de la Queue, et les métriques Cloud.

Les résultats présentés sont issus d'un scénario qui contient 10.000 requêtes Client, extraite à partir du cloud et qui s'étalent sur une durée de 4h.

La figure suivante montre la structure de fichier de résultat des workers. Chaque ligne présente un worker créé au cours de la simulation. Les workers sont triés à la base du temps d'arrêt le plus récent :

id_worker	Running_time	Waiting_time	Switch_On_Date	Switch_Off_Date
5	0j:0H:5m:34s	0j:0H:44m:25s	10/06/2020 11:00:00.000	10/06/2020 11:50:00.000
8	0j:0H:3m:57s	0j:0H:46m:2s	10/06/2020 11:00:00.000	10/06/2020 11:50:00.000
10	0j:0H:1m:46s	0j:0H:48m:13s	10/06/2020 11:00:00.000	10/06/2020 11:50:00.000
11	0j:0H:2m:7s	0j:0H:47m:52s	10/06/2020 11:00:00.000	10/06/2020 11:50:00.000
12	0j:0H:1m:41s	0j:0H:48m:18s	10/06/2020 11:00:00.000	10/06/2020 11:50:00.000
13	0j:0H:1m:57s	0j:0H:48m:2s	10/06/2020 11:00:00.000	10/06/2020 11:50:00.000
15	0j:0H:2m:21s	0j:0H:47m:38s	10/06/2020 11:00:00.000	10/06/2020 11:50:00.000
16	0j:0H:2m:22s	0j:0H:47m:37s	10/06/2020 11:00:00.000	10/06/2020 11:50:00.000
17	0j:0H:1m:39s	0j:0H:48m:20s	10/06/2020 11:00:00.000	10/06/2020 11:50:00.000
18	0j:0H:1m:22s	0j:0H:48m:37s	10/06/2020 11:00:00.000	10/06/2020 11:50:00.000
19	0j:0H:2m:20s	0j:0H:47m:39s	10/06/2020 11:00:00.000	10/06/2020 11:50:00.000
20	0j:0H:2m:38s	0j:0H:47m:21s	10/06/2020 11:00:00.000	10/06/2020 11:50:00.000
24	0j:0H:2m:21s	0j:0H:47m:38s	10/06/2020 11:00:00.000	10/06/2020 11:50:00.000
32	0j:0H:2m:28s	0j:0H:0m:1s	10/06/2020 11:47:30.000	10/06/2020 11:50:00.000
34	0j:0H:0m:22s	0j:0H:1m:47s	10/06/2020 11:47:50.000	10/06/2020 11:50:00.000
35	0j:0H:0m:21s	0j:0H:1m:48s	10/06/2020 11:47:50.000	10/06/2020 11:50:00.000
36	0j:0H:0m:19s	0j:0H:1m:50s	10/06/2020 11:47:50.000	10/06/2020 11:50:00.000
37	0j:0H:0m:24s	0j:0H:1m:45s	10/06/2020 11:47:50.000	10/06/2020 11:50:00.000
38	0j:0H:0m:22s	0j:0H:1m:47s	10/06/2020 11:47:50.000	10/06/2020 11:50:00.000
39	0j:0H:0m:21s	0j:0H:1m:48s	10/06/2020 11:47:50.000	10/06/2020 11:50:00.000
7	0j:0H:6m:49s	0j:0H:45m:10s	10/06/2020 11:00:00.000	10/06/2020 11:52:00.000
14	0j:0H:4m:45s	0j:0H:47m:14s	10/06/2020 11:00:00.000	10/06/2020 11:52:00.000
21	0j:0H:3m:38s	0j:0H:48m:21s	10/06/2020 11:00:00.000	10/06/2020 11:52:00.000
23	0j:0H:3m:52s	0j:0H:48m:7s	10/06/2020 11:00:00.000	10/06/2020 11:52:00.000
25	0j:0H:4m:23s	0j:0H:47m:36s	10/06/2020 11:00:00.000	10/06/2020 11:52:00.000
26	0j:0H:4m:19s	0j:0H:47m:40s	10/06/2020 11:00:00.000	10/06/2020 11:52:00.000
28	0j:0H:3m:41s	0j:0H:48m:18s	10/06/2020 11:00:00.000	10/06/2020 11:52:00.000
30	0j:0H:4m:14s	0j:0H:47m:45s	10/06/2020 11:00:00.000	10/06/2020 11:52:00.000
31	0j:0H:3m:51s	0j:0H:0m:38s	10/06/2020 11:47:30.000	10/06/2020 11:52:00.000
33	0j:0H:2m:35s	0j:0H:1m:54s	10/06/2020 11:47:30.000	10/06/2020 11:52:00.000

Figure 46: Résultat de la simulation - état des Workers

La figure suivante montre la structure du fichier de résultats de la file d'attente. Chaque ligne présente une requête Client traitée par le simulateur :

Queue Time (ms)	Response Time (s)	Worker ID
0	1.77	1
0	6.18	1
0	0.63	2
0	22.31	1
0	22.39	2
0	22.44	3
0	24.25	4
0	22.74	5
0	20.31	6
0	21.83	7
0	22.35	1
0	20.44	2
0	23.93	3
0	22.87	8
0	1.4	9
0	22.37	4
0	0.12	5
0	22.57	5
0	17.74	6
0	21.96	7
0	22.28	1
0	21.29	2
0	20.41	3
0	3.68	8
0	24.03	4
0	20.28	5
0	22.05	6
0	21.69	7
0	0.16	8
0	0.15	1

Figure 47: Résultat de la simulation - état de la queue

La figure suivante montre les différentes métriques de performances présentées par notre système :

Performance Metrics	Values
Max Queue time (s)	27.27s
Average Queue time (s)	0.25s
Requests With No WaitingTime %	95.73%
Total Time no worker available	0j:0H:5m:42s
Workers availability %	98.10%

Figure 48: Résultat de la simulation – Les métriques de performance

La figure suivante montre le diagramme de disponibilité des workers au cours de la simulation :

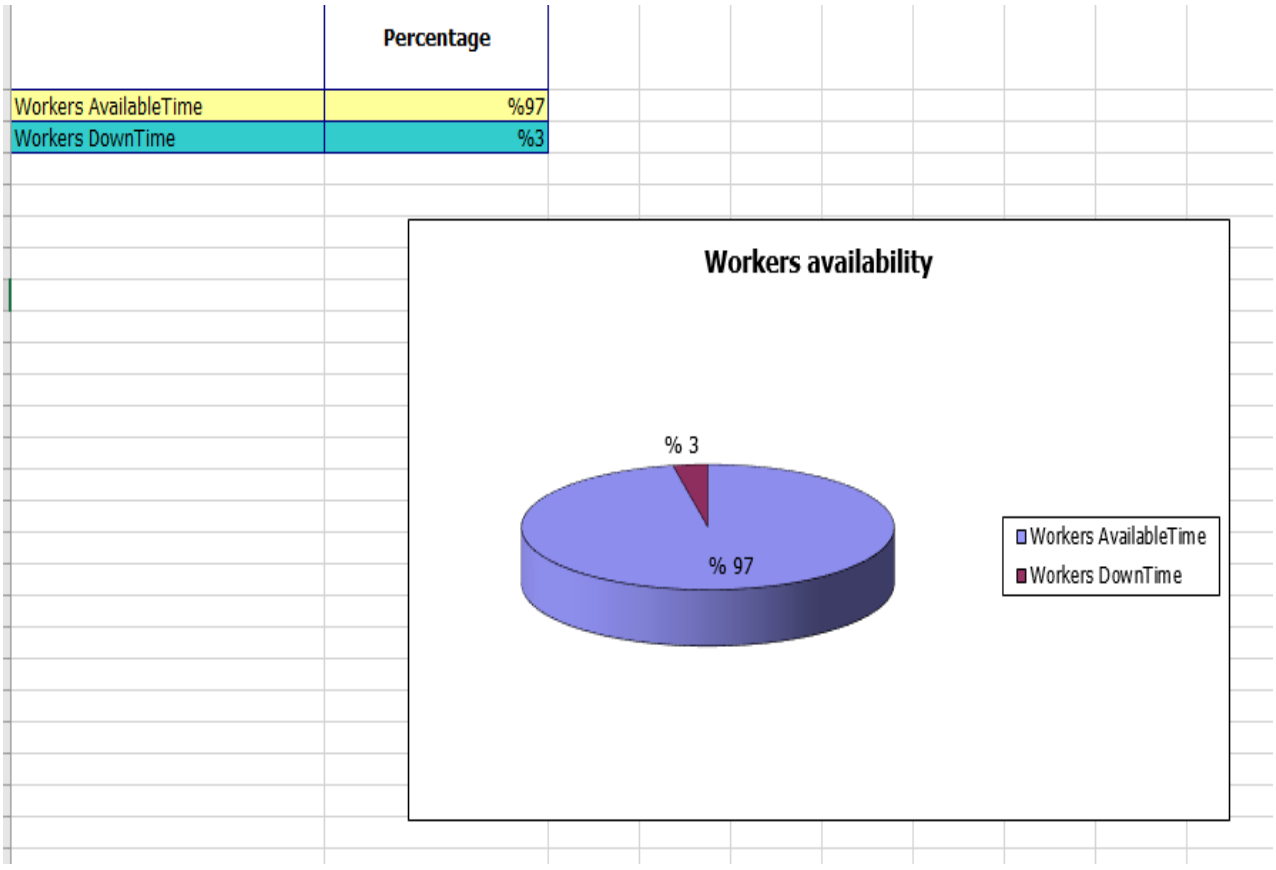

Figure 49: Résultat de la simulation – La disponibilité des workers au cours de la simulation

La figure suivante montre les différentes métriques de facturation présentées par notre système :

Billing Metrics	Value
Configuration Cost($)	33.57 $
workers IdleTime Cost($)	10 $
Total Workers Idle Time	45h
Average Workers Efficiency	33.89%

Figure 50: Résultat de la simulation – Les métrique de facturation

3. Conclusion

Ce dernier chapitre été consacré à la concrétisation de l'étude menée au niveau des chapitres précédents, il présente en bref les différents outils utilisés, il donne ainsi une vue sur la solution développée en présentant la liste des inputs et outputs de notre système.

Conclusion générale

Ce projet de fin d'études a réellement été le fruit d'une expérience très enrichissante. En effet, ce fut l'occasion pour moi de travailler sur un projet réel qui m'a permis de mettre en pratique la méthode agile XP, et de découvrir des nouveaux concepts telles que les services Cloud, l'Auto Scaling, Load Balancing etc.

Ce projet consiste en la réalisation d'un simulateur qui permet de simuler le fonctionnement du service d'auto Scaling de la SG ATS. Ce dernier se basait sur un fichier de configurations générées manuellement, ce qui engendrait souvent soit une sous-estimation des ressources et donc une surcharge de services et un temps de réponse client insatisfaisant ; soit une surestimation de ressources et donc un coût de déploiement très élevé sur le Cloud. Le but de ce stage est d'améliorer l'efficacité du processus du Scaling et d'optimiser les charges de déploiement sur le Cloud.

Dès le début de mon stage, j'ai suivi des formations dans les technologies avec lesquelles j'ai travaillé tout au long de ce projet. J'ai été amené par la suite à étudier la logique métier des différents concepts autour de Service de Scaling.

Pour satisfaire les exigences du projet, j'ai eu recours à la méthode agile XP, pour assurer un bon suivi du projet et garantir le développement d'une solution solide.

Le projet a été planifié en sept itérations basées sur sept scénarios principaux pour une durée de cinq mois.

Les objectifs exprimés dans le cahier de charges sont tous atteints et les responsables de la SG ATS sont satisfaits des résultats que j'ai obtenus.

J'ai réalisé une solution qui permet d'améliorer la qualité de déploiement des services exposés sur le cloud avec la possibilité de mesurer la satisfaction du Client Cloud et de calculer le coût du Fournisseur Cloud.

J'ai pu simuler le fonctionnement du service de Scaling et obtenir les résultats attendus pour chaque service déployé sur le Cloud :

- L'état des Workers de chaque configuration du service Cloud ;
- L'état des requêtes Client ;
- Les métriques de qualité.

Parmi les principales difficultés auxquelles j'ai dû faire face dans la réalisation de ce projet, c'était la pandémie du Covid19 qui m'a poussé à suspendre le travail durant tout un mois. Aussi l'unicité du projet et sa technicité, ont rendu son explication dans un cadre pédagogiques, une tache un peu compliquée.

Comme extensions possibles à ce projet, plusieurs améliorations peuvent être considérées. Nous pouvons par exemple penser à ajouter un module qui permet de comparer

deux configurations en se basant sur les résultats de la simulation de chacune d'entre elles. Nous pouvons aussi aller plus loin, et étendre le système pour qu'il puisse générer une configuration optimale en se basant seulement sur une charge de requêtes donnée en input.

Finalement et comme il s'agit d'un projet extensible, nous avons conçu et développé cette solution d'une façon modulaire en évitant le fort couplage entre les composants pour laisser place à toute future extension possible.

Webographie

https://africa-technologies-services.sgcib.com/fr/about/

https://azure.microsoft.com/en-us/overview/what-is-iaas/

https://azure.microsoft.com/en-us/overview/what-is-paas/

https://azure.microsoft.com/en-us/overview/what-is-saas/

https://en.wikipedia.org/wiki/Simulation_software

https://en.wikipedia.org/wiki/Discrete-event_simulation

https://git-scm.com/doc

https://cloud.google.com/compute/docs/autoscaler?hl=fr

https://stackoverflow.com/

https://docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/web-queue-worker

https://en.wikipedia.org/wiki/Continuous_simulation

https://azure.microsoft.com/fr-fr/overview/what-is-cloud-computing/

https://fr.wikipedia.org/wiki/Git

https://fr.wikipedia.org/wiki/Extreme_programming

https://docs.microsoft.com/en-us/dotnet/csharp/roslyn-sdk/

https://fr.wikipedia.org/wiki/visualStudio

https://searchcloudcomputing.techtarget.com/definition/Infrastructure-as-a-Service-IaaS

https://www.w3.org/wiki/CloudComputing

https://www.amazon.com/Extreme-Programming-Explained-Embrace-Change/dp/0321278658

https://docs.microsoft.com/fr-fr/dotnet/framework/

https://www.jetbrains.com/resharper/

Les annexes

Dans ce qui suit seront présentés les documents annexes proposant plus de détails sur quelques concepts et définitions utilisés pour concevoir et réaliser ce projet.

Annexe 1 : Concept d'Auto Scaling

1) Qu'est-ce que la mise à l'échelle automatique ?

La mise à l'échelle automatique est un moyen d'augmenter ou de réduire automatiquement le nombre de machines allouées à votre application en fonction de ses besoins à tout moment.

Avant le cloud computing, il était très difficile de faire évoluer un site Web, et encore moins de trouver un moyen de mettre automatiquement à l'échelle (mise à l'échelle automatique) une configuration de serveur. Dans un environnement d'hébergement traditionnel et dédié, vous êtes limité par vos ressources matérielles. Une fois que ces ressources de serveur sont maximisées, votre site souffrira inévitablement du point de vue des performances et risque de planter, ce qui vous fera perdre des données et / ou des activités potentielles. L'auto scaling permet de créer une configuration automatisée qui réagit automatiquement aux diverses conditions surveillées lorsque les seuils sont dépassés.

Aujourd'hui, le cloud computing révolutionne totalement la façon dont les ressources informatiques sont allouées, permettant de construire une configuration de serveur entièrement évolutive sur le Cloud. Nous avons désormais la possibilité de lancer des ressources de calcul supplémentaires à la demande et de les utiliser aussi longtemps que vous le souhaitez, puis de les arrêter lorsqu'elles ne sont plus nécessaires.

2) Principes de base

I. Les groupes d'instances

Un groupe d'instances est un ensemble d'instances de machines virtuelles (VM) qui peuvent être gérées en tant qu'entité unique.

II. Règles d'autoscaling et utilisation cible

Pour créer un autoscaler, vous devez spécifier la règle d'autoscaling et un niveau d'utilisation cible que l'autoscaler utilise pour déterminer à quel moment faire évoluer le groupe. Vous pouvez choisir de faire évoluer votre groupe à l'aide des règles suivantes :

- Utilisation moyenne du processeur
- Capacité de diffusion de l'équilibrage de charge HTTP, qui peut être basée sur l'utilisation ou le nombre de requêtes par seconde
- Métriques Cloud Monitoring

L'autoscaler collecte en permanence les informations d'utilisation en fonction de la règle, compare l'utilisation réelle à l'utilisation cible souhaitée et détermine si la taille du groupe doit être augmentée ou réduite.

Le niveau d'utilisation cible est le niveau auquel vous souhaitez maintenir vos instances de machine virtuelle (VM). Par exemple, si vous faites évoluer vos instances en fonction de l'utilisation du processeur, vous pouvez définir votre niveau d'utilisation cible à 75 % pour que l'autoscaler maintienne l'utilisation du processeur du groupe d'instances spécifié à 75 % ou presque. Le niveau d'utilisation de chaque métrique est interprété différemment selon la règle d'autoscaling.

III. Intervalle entre chaque période d'autoscaling

Lorsqu'une instance est en cours d'initialisation, les informations concernant son utilisation peuvent ne pas correspondre aux circonstances normales. Par conséquent, ces informations peuvent ne pas être fiables pour les décisions de l'autoscaler. Il est donc préférable de ne pas les prendre en compte. Il faut Spécifier un intervalle pour que les instances puissent finir de s'initialiser avant que l'autoscaler ne commence à collecter leurs informations.

Les délais d'initialisation réels peuvent varier en fonction de plusieurs facteurs. Pour cela il faut tester le temps nécessaire à l'initialisation de l'application. Pour ce faire, on peut créer une instance et mesurez la durée du processus de démarrage à partir du moment où l'instance passe à l'état RUNNING jusqu'à ce qu'elle soit prête.

Si nous définissons un intervalle entre chaque période d'autoscaling nettement plus courte que le temps d'initialisation, l'autoscaler risque d'ignorer les données d'utilisation légitimes et de sous-estimer la taille requise pour le groupe.

IV. Période de stabilisation

Afin de supprimer des instances, l'autoscaler calcule la taille cible recommandée du groupe en fonction de la charge maximale des 10 dernières minutes (c'est paramétrable). On parle pour ces 10 dernières minutes de période de stabilisation.

Annexe 2 : Architecture Azure Cloud Web-Queue-Worker

Un style d'architecture est une famille d'architectures qui partagent certaines caractéristiques. Par exemple, N-tier est un style d'architecture courant. Plus récemment, les architectures de microservices ont commencé à gagner du terrain. Les styles d'architecture ne nécessitent pas l'utilisation de technologies particulières, mais certaines technologies sont bien adaptées à certaines architectures. Par exemple, les conteneurs conviennent parfaitement aux microservices.

Web-Queue-Worker est une architecture Cloud Azure, qui permet de déployer des services Cloud.

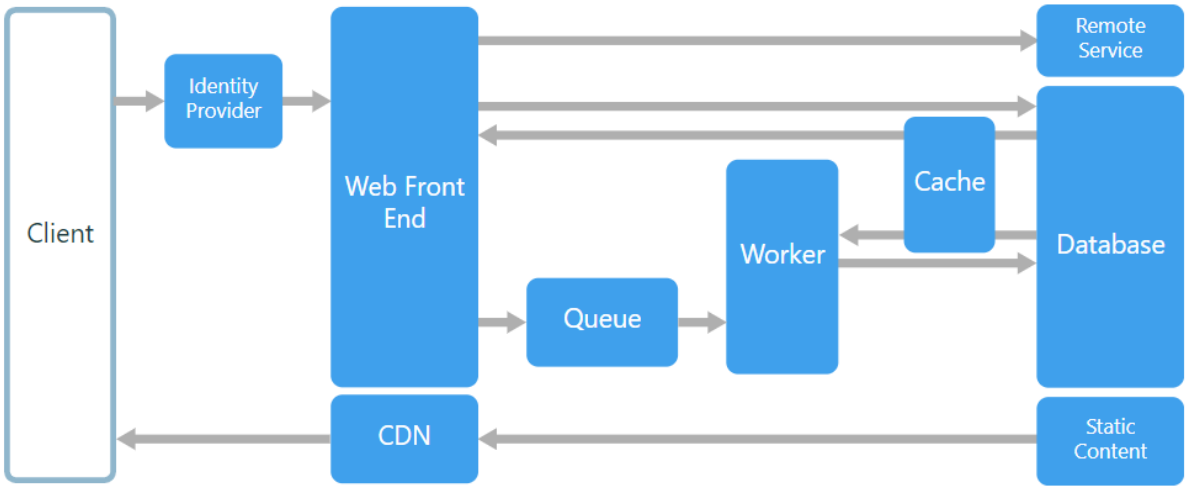

Figure 51: Architecture Azure web-queue-worker

1) Les principaux composants de l'architecture Web-Queue-Worker

Les principaux composants de cette architecture sont un frontal Web qui répond aux demandes des clients et un travailleur qui effectue des tâches gourmandes en ressources, des flux de travail de longue durée ou des travaux par lots. Le serveur Web frontal communique avec le travailleur via une file d'attente de messages.

Les autres composants généralement intégrés à cette architecture sont les suivants :

- Une ou plusieurs bases de données.
- Un cache pour stocker les valeurs de la base de données pour des lectures rapides.
- CDN pour diffuser du contenu statique
- Services à distance, tels que le courrier électronique ou le service SMS. Ils sont souvent fournis par des tiers.
- Fournisseur d'identité pour l'authentification.

2) Utilisation de cette architecture

L'architecture Web-Queue-Worker est généralement implémentée à l'aide de services de calcul gérés, soit Azure App Service ou Azure Cloud Services.

Considérez ce style d'architecture pour :

- Applications avec un domaine relativement simple.
- Applications avec des workflows de longue durée ou des opérations par lots.
- Lorsque vous souhaitez utiliser des services gérés, plutôt que l'infrastructure en tant que service (IaaS).

3) Avantages

- Architecture relativement simple et facile à comprendre.
- Facile à déployer et à gérer.
- Séparation claire des préoccupations.
- La partie frontale est découplée du travailleur à l'aide de la messagerie asynchrone.
- Le frontal et le travailleur peuvent être mis à l'échelle indépendamment.

Annexe 3 : L'Extreme Programming

1) Les valeurs de bases d'eXtreme Programming

Pour que les pratiques de l'eXtreme Programming fonctionnent, il est nécessaire de respecter 4 valeurs. (Bénard,2005)

I. La communication pour une meilleure visibilité

La communication est en effet un point indispensable dans une équipe XP. Lorsqu'on parle de communication, il s'agit surtout de communication directe (orale). Elle favorise en effet le transfert de compétence, la détection de problème, l'entraide au sein de l'équipe et la confiance entre le client et l'entreprise.

II. La simplicité comme garantie de productivité

Côté client, c'est à lui de définir ce dont il a réellement besoin et d'y mettre des priorités. Côté développeur, il est inutile de s'occuper de cas qui n'arriveront peut-être jamais. C'est en appliquant cette simplicité que la rapidité de développement et la possibilité de changement seront favorisées.

III. Le feedback comme outil de réduction du risque

Le feedback est au coeur de la méthode eXtreme Programming. En effet, elle est gage de qualité. On retrouve la notion de feedback à différentes échelles du projet : des tests unitaires aux tests de recettes jusqu'à la structure itérative de la méthode.

IV. Le courage de prendre les bonnes décisions

Le courage est une qualité qu'il faut pour pouvoir respecter les trois précédentes valeurs. Il faut du courage pour se focaliser uniquement sur ce dont on a besoin, de maintenir une transparence en communiquant sur nos difficultés (et nos atouts) et admettre les différents feedbacks comme par exemple devoir jeter une partie de code qu'on aurait créé et qui ne conviendrait pas.

2) Les règles d'eXtreme Programming

Il existe 13 règles si on veut respecter l'eXtreme Programming. Celles-ci se découpent en 3 catégories :

I. Les pratiques de programmation

- Une conception simple (simple design) : inutile d'inventer des mécanismes génériques, le "péché mignon" du développeur, si ce n'est pas nécessaire. Il faut toujours implémenter la solution la plus simple.
- Le remaniement (refactoring) : Il est important que les développeurs reviennent sur leur code pour le rendre plus simple, enlever les parties inutilisées.
- Développement piloté par les tests unitaires (Test Driven Developpement) : Le développeur écrit des tests unitaires avant d'écrire le code. Cette pratique lui permet de bien structurer dans son esprit ce qu'il veut faire ainsi que d'éviter le phénomène de régression.
- Test de recette (acceptance test) : Ces tests sont créés par le client. Il lui permet de définir les objectifs que les développeurs doivent atteindre. Si le test passe, alors la fonctionnalité est validée tel que le client la veut.

II. Les pratiques de collaboration

- Programmation en binôme (pair programming) : Les développeurs doivent développer en binôme (deux par ordinateur). Cela permet une relecture en temps réel du code. Les binômes sont changés régulièrement
- Responsabilité collective du code (collective code ownership : Les développeurs travaillent en binôme et changent régulièrement, ainsi, chacun travaille sur chaque partie de l'application.
- Règle de codage (coding standards) : Des règles de codage sont décidées au début du projet. Chacun doit s'y tenir afin de garantir la cohérence du code et ainsi faciliter la modification de celui-ci par n'importe quel développeur.
- Métaphore : Le système ainsi que les sous-systèmes doivent être décrit par des métaphores ceci dans le but de faciliter la communication.
- Intégration continue (continuous integrated) : Les développeurs doivent synchroniser le plus souvent possible leur travail afin de faciliter le processus d'intégration.

III. Les pratiques de gestion de projet

- Livraisons fréquentes (fréquent releases) : Le logiciel doit être livré régulièrement. Cela permet au client ainsi qu'aux développeurs de s'assurer que le produit correspond bien aux attentes.
- Planification itérative (planning game) : Lors d'une réunion dédiée, le planning décrivant l'itération en cours est décidé par le client et l'équipe de développement.
- Client sur site (on-site customer) : Le client doit être intégré à l'équipe, la communication est ainsi optimale. Le client au sens eXtreme Programming n'est pas forcément le client au sens commun. Le client, au sens XP, est celui qui prend les décisions et qui est capable de répondre aux questions des développeurs.
- Rythme durable (sustainable pace) : L'équipe de développement doit avoir un rythme qui peut être conservé tout le long du projet. Ce rythme doit favoriser le travail de qualité. Les périodes de "rush" sont à éviter.

3) Les rôles de l'eXtreme Programming

eXtreme Programming définit six rôles principaux dans un projet IT :

- **Le programmeur** : c'est le développeur. Dans son rôle XP, il est responsabilisé. En effet, sa proximité avec le client ainsi que le respect des différentes règles (dont la responsabilité collective du code) responsabilisent et revalorise le métier client au sens XP est celui qui est présent au sein de l'équipe de développement, qui répond aux questions des développeurs et définit la priorité dans laquelle doit être implémenté les fonctionnalités. Il a le devoir de fournir un feedback aux développeurs ainsi que des tests de recette.
- **Le testeur** : le testeur est le bras droit du client. C'est à lui que revient la tâche d'installer les outils nécessaires à l'intégration continus, des tests de recettes et unitaires. Il est également là pour aider le client à créer les tests de recette.
- **Le tracker** : Le tracker est là pour suivre l'avancement des tâches au sein d'une itération. Son rôle est donc de détecter au plus vite les éventuelles difficultés qui surviennent afin de les résoudre le plus vite possible. Ce rôle doit être évité d'être tenu par un membre hiérarchique supérieur afin d'empêcher un sentiment de surveillance.
- **Le manager** : C'est le supérieur hiérarchique des développeurs. Il est là pour demander des comptes (à son équipe XP comme au client) et fournir les moyens humains et matériels pour l'avancement du projet. Il ne participe qu'aux réunions de planification et doit donc faire confiance à son équipe XP pour la réalisation du projet.
- **Le coach** : Le coach est le garant du respect des règles eXtreme Programming. C'est à lui que revient le rôle de mettre en place la méthode. Son objectif est que l'équipe puisse fonctionner sans lui. Il doit donc s'armer de courage pour dire les choses telles qu'elles sont.

Un des objectifs de toutes ces règles est de pouvoir faire face aux changements réguliers d'un projet IT. Voici une évolution que l'on retrouve souvent dans les projets IT :

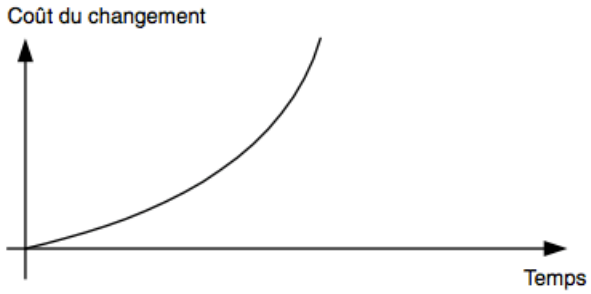

Figure 1-2. Évolution supposée du coût du changement dans un logiciel

Et voici la courbe du coût du changement revue par les auteurs d'eXtreme Programming :

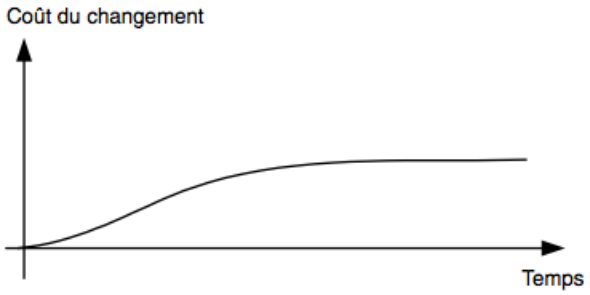

Figure 1-3. L'évolution du coût du changement dans le logiciel revue par les auteurs d'XP

4) Facteur de compétitivité des entreprises

Dans une première partie, nous expliquons la façon d'amener la méthode eXtreme Programming. En effet, certaines pratiques d'eXtreme Programming peuvent être vues par l'entreprise comme une perte de rentabilité (la programmation en binôme) ou alors comme du "flicage" pour les développeurs (le rôle de tracker par exemple).

La formation de toute l'équipe à l'eXtreme Programming est indispensable. Il est d'ailleurs important de souligner que la philosophie d'eXtreme Programming insiste dans le fait que toute l'équipe doit participer aux diverses formations. 3 différentes méthodes sont communiquées :

- Le coaching : un consultant eXtreme Programming est présent dans l'entreprise et joue le rôle de coach.
- L'immersion XP : L'équipe se rend à une formation sur plusieurs jours et est encadrée par une entreprise de formation dans la création d'un projet fictif en utilisant les pratiques eXtreme Programming.
- L'Heure Extreme : Ce sont des ateliers d'environ une heure mettant en pratique une notion d'eXtreme Programming.

Toutes les pratiques eXtreme Programming peuvent être dures à mettre en place dans une équipe. Il est donc recommandé de commencer par certaines pratiques comme la planification itérative et l'élaboration des tests unitaires avant le code.

5) Coûts et retours sur investissement

Cette partie décrit les conséquences et la manipulation des variables clés qui font d'un projet XP une réussite.

L'EXtreme Programming identifie 4 variables clés dans le déroulement d'un projet :

- Les coûts directs et indirects
- La qualité livrée
- La durée des projets
- Le périmètre fonctionnel des projets

Suite à cette analyse, la seule variable qui ressort manipulable est le périmètre fonctionnel des projets. En effet, il est impensable de diminuer la qualité du produit. Il est possible de modifier les coûts et le temps de développement mais le client ne sera pas satisfait. Par contre, réduire le nombre de fonctionnalités d'une version pour conserver la qualité, le coût et la durée de développement sera plus envisageable par le client.

6) *Les aspects contractuels*

L'EXtreme Programming pose une certaine problématique lors de l'aspect contractuelle. En effet, la méthode n'est pas applicable suivant le type de contrat. En France, il existe 3 types de contrats :

- Les contrats forfaitaires : Le client définit un cahier des charges précis et contactera une société de service pour le mettre en oeuvre.
- Les contrats d'assistance technique : Le client définit et met en oeuvre le cahier des charges en faisant appel à du personnel compétent chez un fournisseur.
- Les contrats d'assistance forfaitée : Le client contacte un fournisseur qui lui fournira une équipe entière chargé de la réalisation du projet.

Seul le dernier contrat est véritablement adapté à la méthode XP (bien qu'il soit possible de l'appliquer au second). En effet, le fait que ce soit une équipe en charge de la réalisation et d'être chez le client, le déploiement de la méthode est favorisé. Cependant il va falloir adapter les termes du contrat pour pouvoir spécifier les principes d'XP tels que le travail en binôme et le rythme durable par exemple. Le droit d'arrêt du projet est également donné au client, la méthode XP, grâce à sa structure itérative ainsi qu'aux livraisons d'un logiciel fonctionnel favorise la facture de ce qui est déjà fait.

7) *Qualité et processus*

La norme ISO:9000 est définie telle :

- L'orientation client
- Leadership
- Implication du personnel
- Approche processus
- Management par approche système
- Amélioration continue
- 7. Approche factuelle pour la prise de décision 8. Relations mutuellement bénéfiques avec les fournisseurs

Nous expliquons alors les convergences entre l'eXtreme Programming et cette norme ainsi que les différents processus tel que les indicateurs de qualité (tests unitaires, recette) qui inscrivent XP dans la norme ISO:9001.

8) Méthodologie

La méthode eXtreme Programming s'inscrit dans un mouvement plus vaste : celui de l'Agilité. Pour qu'une méthode soit validée comme étant agile, elle doit respecter le manifeste Agile :

L'importance est donnée:

- Aux individus et aux interactions plus qu'aux processus et aux outils
- À des logiciels immédiatement disponibles plus qu'à une documentation exhaustive
- À la collaboration avec le client plus qu'à la négociation contractuelle
- À la réactivité face au changement plus qu'au respect d'un plan

Les méthodes Agiles s'appuient sur ces 4 valeurs clés et se différencient par leurs principes et leurs pratiques.