Test des performance

4e année ingénierie de web

Azulance

ACHBOUQ Slimane
ALAA HAZIM Zaid
ATTAOUI Zakaria

Professeur: **Banaei** Mohammad-amine

Etablissement : Ecole supérieure en génie informatique **ESGI**
Promotion : 2021 - 2022

Performance Test Plan Sign-off

Table 1: Sign-off Detail

Name	Role / Designation	Signoff Date	Signature
ALAA HAZIM Zaid	Web Developer		
ACHBOUQ Slimane	Web Developer		
ATTAOUI Zakaria	Web Developer		

Record of Changes

< Provide information on how the development and distribution of the performance test plan were carried out and tracked with dates. Use the table below to provide the version number, the date of the version, the author/owner of the version, and a brief description of the reason for creating the revised version.>

Table 2: Record of Changes

Version Number	Date	Author/Owner	Description of Change
Draft	01/01/2019	PerfMatrix	Draft version with available details
0.1	15/01/2019	PerfMatrix	Added NFR details
0.2	30/01/2019	PerfMatrix	Added Environment details
xx	xx/xx/xxxx	xxxxxx	xxxx xxxx xxxx xxx

Table of Contents

1. Executive Summary

Le test de performance va permettre de déterminer le bon fonctionnement de la plateforme et de s’assurer que la performance répond à ce qui est attendu.

1.1 Overview: Project Background and Scope

L'application qui sera testée est une plateforme d'assurance pour véhicules (voiture + moto). Un client peut se créer un compte, souscrire à une ou plusieurs assurances et consulter ses dossiers. L'objectif du site est de proposer des assurances adaptées à tout véhicule deux roues et quatre roues, pour le plus de profils clients possible. Tout type d'utilisateurs est prévu, à défaut d'avoir la majorité pour pouvoir souscrire une assurance.

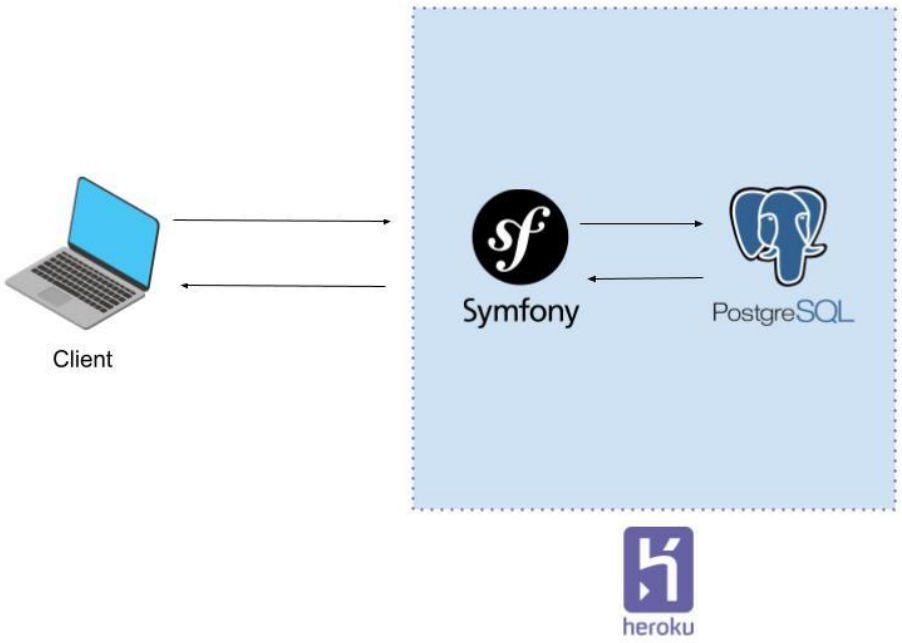
2. Application Architecture

L'application est développée en PHP sous Symfony en architecture monolithique. Le front est réalisé en Twig et avec Bootstrap.

2.1 Overview: System Architecture

La partie Back-end est développée en PHP sous Symfony et en utilisant donc une multitude de librairies (login, système de paiement, Wizard Form, ..). La partie Front-end est réalisée en Twig sous Symfony et en utilisant la bibliothèque Bootstrap qui fournit une multitude de feuilles CSS. L'infrastructure de production utilisée est Heroku, une plateforme gratuite d'hébergement de sites web.

2.2 Architecture Diagram



2.3 Detailed information on each component

Il y a plusieurs fonctionnalités qui pourraient avoir un impact sur le site. Il y a la partie utilisateurs où une création de compte est possible, un système de paiement qui survient après la finalisation d'un dossier d'assurance, ainsi qu'un formulaire de contact pour envoyer un message à un administrateur de la plateforme. Ainsi une

bonne performance est recommandée pour ces fonctionnalités afin d’assurer un bon fonctionnement.

3. Performance Test Requirements

3.1 Requirements

Il est important de réaliser des tests de performance sur ce projet afin d'avoir une expérience utilisateur fluide et donc s'assurer que la performance souhaitée est bien présente.

3.1.1 Business NFR

Table 3: Business NFR

Business Transactions	User Load	SLA/response times	Transactions per hour
Logged users	50	2 seconds	100
Paiement	30	3 seconds	200
Register	40	3 seconds	100
Dossiers assurance	50	2 seconds	300

3.2 Detailed and Agreed NFR

NFR 1 : Lors d'une authentification d'utilisateur, il faut que la fonctionnalité mette au maximum 3 secondes.

NFR 2 : Lors d'un paiement, il faut que la fonctionnalité mette au maximum 10 secondes.

NFR 3 : Lors d'une création de compte, il faut que la fonctionnalité mette au maximum 3 secondes.

NFR 4 : Lors d'une finalisation de création d'un dossier, il faut que la création prenne au maximum 5 secondes.

3.3 NFR and NFT Matrix

<This section contains the non-functional test cases (scripts) and applicable non-functional requirement>

Table 4: NFR-NFT Matrix

	NFT1	NFT2	NFT3	NFT4	NFT5
NFR1	×	√	×	×	×
NFR2	×	×	√	×	×
NFR3	√	×	×	√	×
NFR4	√	×	×	×	×

NFR5	×	√	√	×	√

4. Performance Test Planning

4.1 Performance Test Approach

1) **Limitations :** L'infrastructure d'hébergement utilisée est Heroku. C'est une plateforme qui fournit des performances minimales pour faire fonctionner un site de petite envergure.

2) **Modèle de charge :** Nous prévoyons une charge de 50 personnes maximum avec une arrivée progressive.

3) **Type de test nécessaire :** Le type de test à réaliser est Load Testing.

4) **Métriques à surveiller :** Response time, Maximum active seconds, Amount of connection pooling, Memory use

5) **Métriques qui définissent la réussite ou échec :** Response time

Environnement dans lequel vont se dérouler les tests :

1) **CPU, Mémoire :** 8 CPU, 8GO
2) **OS :** Ubuntu
3) **Software pertinent :** compatible

Table 5: Change Requests (CRs)

Task ID	Description	Project Affected
CRNFT01	Response Time too high for Login Page	XXXXXX
CRNFR02	XXXXX	XXXXXX
CRNFT03	XXXXX	XXXXXX

4.1.1 Performance Testing and Monitoring Tool Details

Table 6: Description of Performance Testing Tool

Tool Name	Description	Licensed / Open-Source?	No. of licenses
Micro Focus Performance Center	Version: 12.55	Licensed	10,000

	Required Protocol: Web HTTP/HTML Support Forum Link: Support ID:		
Dynatrace	Version 1.1 Support Forum Link: Support ID:	Licensed	NA
xxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxx

4.1.2 Performance Test Script Steps

<In this section, the performance test scripts that need to be developed are detailed by user action step as shown in the tables below. For each key Business Process within the Application under Test which was agreed from the project; a Performance Test script needs to be developed.

The transaction flow and script details must be given like below table: Develop performance test scripts that simulate all of the actions in the Business Processes/Transactions documented in the Load Model.>

Table 7: Performance Test (Script 1 Steps)

Step #	**Application Name: Azulance** **Business Process Name: Création de compte** **NFT Script Name: 01_Azulance_CreationCompte**
1	Home Page
2	Register
3	Login
4	Logout

Table 8: Performance Test (Script 2 Steps)

Step #	Application Name: Azulance Business Process Name: Paiement NFT Script Name: 01_Azulance_Paiement
1	Home Page
2	Login
3	Formulaire dossier
4	Choisir un tarif
5	Payer
6	Logout

Table 9: Performance Test Runtime Settings (Optional Information, provide only if available)

Script #	**Pacing between Iterations**	**Think Time between transactions**
Script 1	20 seconds (Fixed)	3 seconds (Fixed)

Script 2	3-4 minutes (Random)	10-15 seconds (Random)

4.1.3 Performance Test Data Planning

Pour le premier script, il faut avoir une adresse mail et un mot de passe. Il faut aussi se servir de la boîte mail de l'adresse mail qui sera utilisée car elle recevra un lien de validation de compte, permettant la connexion par la suite.

Pour le second script, il faut saisir une adresse mail et un mot de passe pour se connecter. Ensuite il faut démarrer la création d'un dossier en saisissant des informations à propos du conducteur et du véhicule à assurer (date d'obtention du permis, chevaux du véhicule, etc).

A la fin du parcours, il faut choisir une formule d'assurance sur trois options.

Après avoir choisi une formule, il faut se munir de sa carte bancaire afin de saisir les informations de paiement et valider le paiement.

4.1.3.1 Data Preparation

Il est possible de générer aléatoirement des données qui seront à saisir pour faciliter la saisie (générateur de mot de passe, générateur de mots-clés, jeu de données déjà prêt, ..).

Il est aussi envisageable de préparer des jeux de données.

5. Performance Test Execution

5.1 Performance Test Summary

Table 10: Performance Test Scenarios

Test Run	Test Scenario Summary
Login	Valider la performance du login
Cycle 1 - Run 1	Load Test - 30 min de test avec peak load
Cycle 1 - Run 2	Repeat Load Test - 30 min de test avec peak load
Cycle 1 - Run 3	Repeat Load Test - 40 min de test avec 150% de peak load
Cycle 2 - Run 1	Load Test - 1 heure de test avec peak load
Cycle 2 - Run 2	Repeat Load Test - 1 heure de test avec peak load
Cycle 2 - Run 3	Load test - 2 heures de test avec 200% de peak load
Cycle 2 - Run 4	Repeat Load Test - 2 heures de test avec 200% de peak load

Test Run	Test Scenario Summary
Register	Valider la performance de la création de compte
Cycle 1 - Run 1	Load Test - 30 min de test avec peak load
Cycle 1 - Run 2	Repeat Load Test - 30 min de test avec peak load
Cycle 1 - Run 3	Repeat Load Test - 40 min de test avec 150% de peak load
Cycle 2 - Run 1	Load Test - 1 heure de test avec peak load
Cycle 2 - Run 2	Repeat Load Test - 1 heure de test avec peak load
Cycle 2 - Run 3	Load test - 2 heures de test avec 150% de peak load
Cycle 2 - Run 4	Repeat Load Test - 2 heures de test avec 175% de peak load

Test Run	Test Scenario Summary
Paiement	Valider la performance du système de paiement
Cycle 1 - Run 1	Load Test - 30 min de test avec peak load
Cycle 1 - Run 2	Repeat Load Test - 30 min de test avec peak load
Cycle 1 - Run 3	Repeat Load Test - 40 min de test avec 150% de peak load
Cycle 2 - Run 1	Load Test - 1 heure de test avec peak load
Cycle 2 - Run 2	Repeat Load Test - 1 heure de test avec peak load
Cycle 2 - Run 3	Load test - 2 heures de test avec 125% de peak load
Cycle 2 - Run 4	Repeat Load Test - 2 heures de test avec 150% de peak load

Test Run	Test Scenario Summary
Dossier assurance	Valider la performance de la création d'un dossier d'assurance
Cycle 1 - Run 1	Load Test - 30 min de test avec peak load
Cycle 1 - Run 2	Repeat Load Test - 30 min de test avec peak load
Cycle 1 - Run 3	Repeat Load Test - 40 min de test avec 200% de peak load
Cycle 2 - Run 1	Load Test - 1 heure de test avec peak load
Cycle 2 - Run 2	Repeat Load Test - 1 heure de test avec peak load
Cycle 2 - Run 3	Load test - 2 heures de test avec 150% de peak load
Cycle 2 - Run 4	Repeat Load Test - 2 heures de test avec 200% de peak load

5.2 Performance Test Details

5.2.1 Load Test

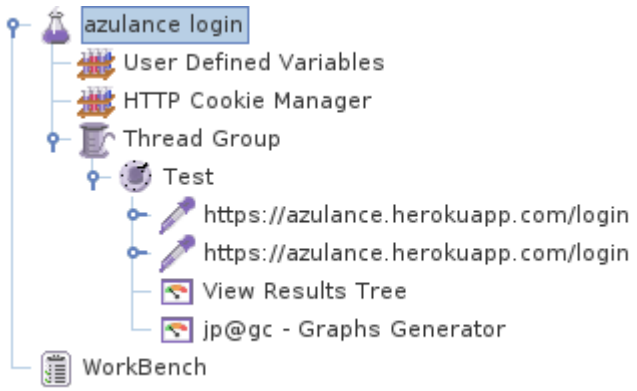
Table 11: Load Test Scenarios Detail

	Test Details
Test ID	Cycle 1-Run1, Cycle 1-Run2, Cycle 1-Run3, Cycle 2-Run1, Cycle 2-Run2 et Cycle 2-Run3
Purpose	Nous testons si la charge qu'on donne au site (et donc à la fonctionnalité concernée) sera supportée, avec une charge haute et pendant une longue durée. Ce test validera les métriques définies.
No. of Tests	6 (3 tests par cycle)
Duration	Ramp-up: 5 secondes Steady State: Ramp-down: 5 secondes
Scripts	1. Login 2. Register 3. Paiement 4. Dossier assurance
Scenario Name	Load Test Scenario
Covered NFR	NFR 1, NFR 2, NFR 3, NFR 4
User Load / Volume	50 Vusers (Threads) Load
Entry Criteria	1. Stabilité de la fonctionnalité 2. Stabilité du site 3. Correspondre aux NFR définis
Exit Criteria	1. Le moins d'erreurs possible 2. Ne pas faire crash Heroku

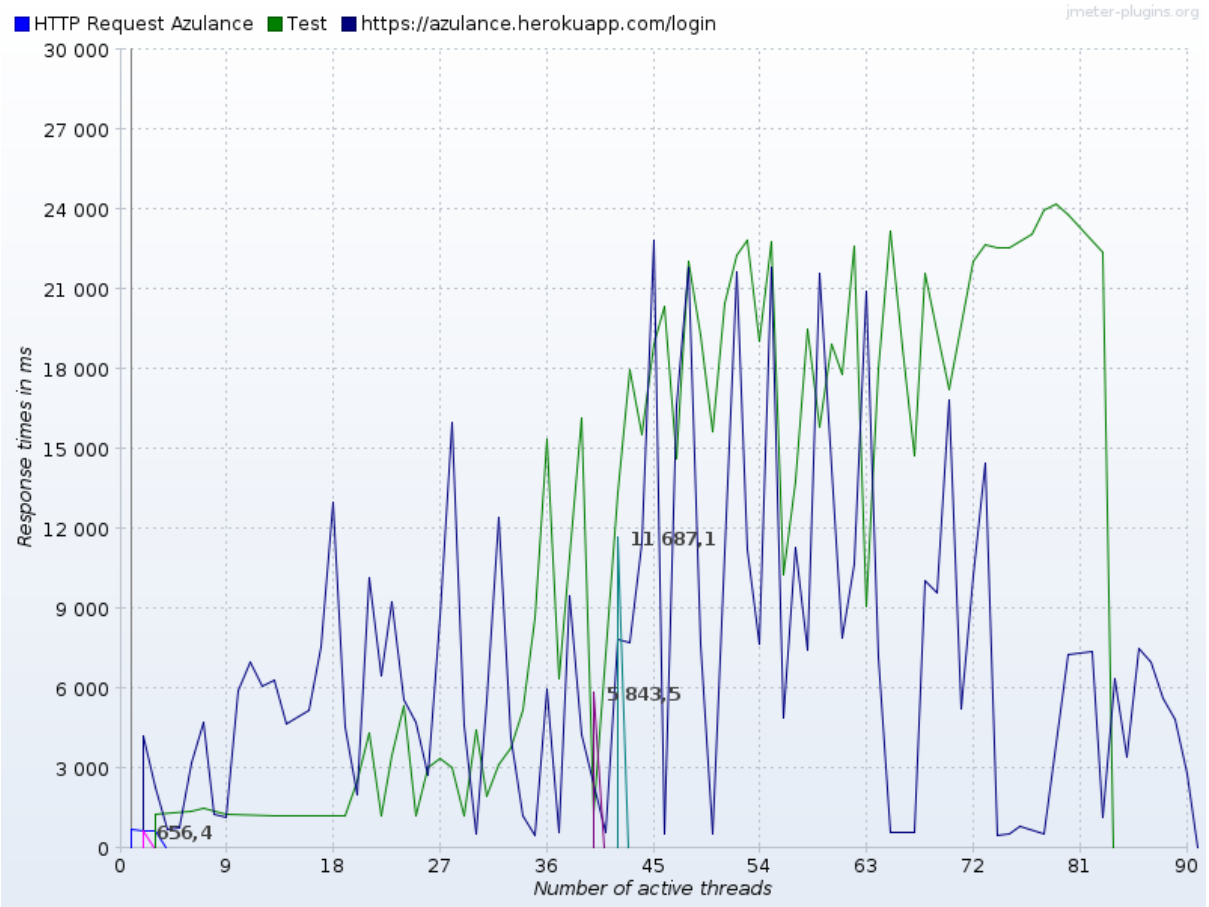
6. Test results

TEST 1 : Résultats de test réalisé sur le scénario de Login, en se basant sur le Cycle 2 Run 4 pour tester les limites, sachant que nous utilisons l'hébergeur gratuit Heroku.

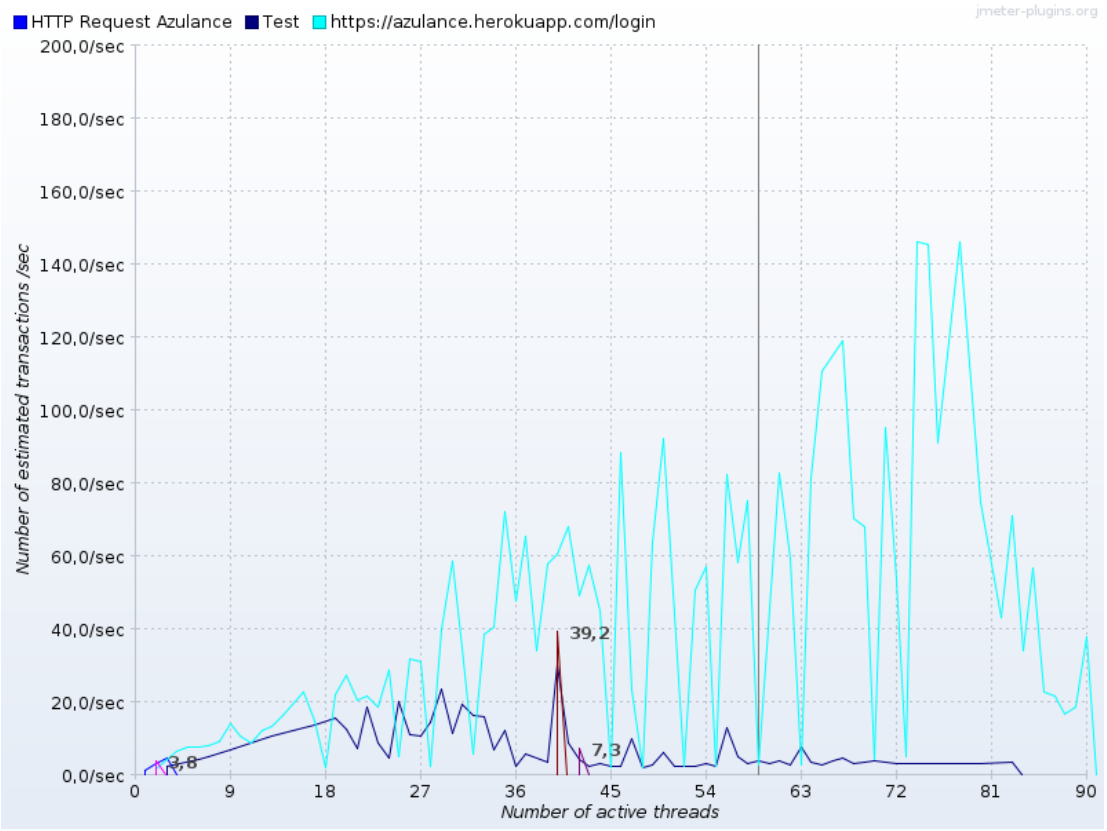
Nous avons d'abord généré un fichier .jmx à l'aide de BlazeMeter et avons gardé l'essentiel :



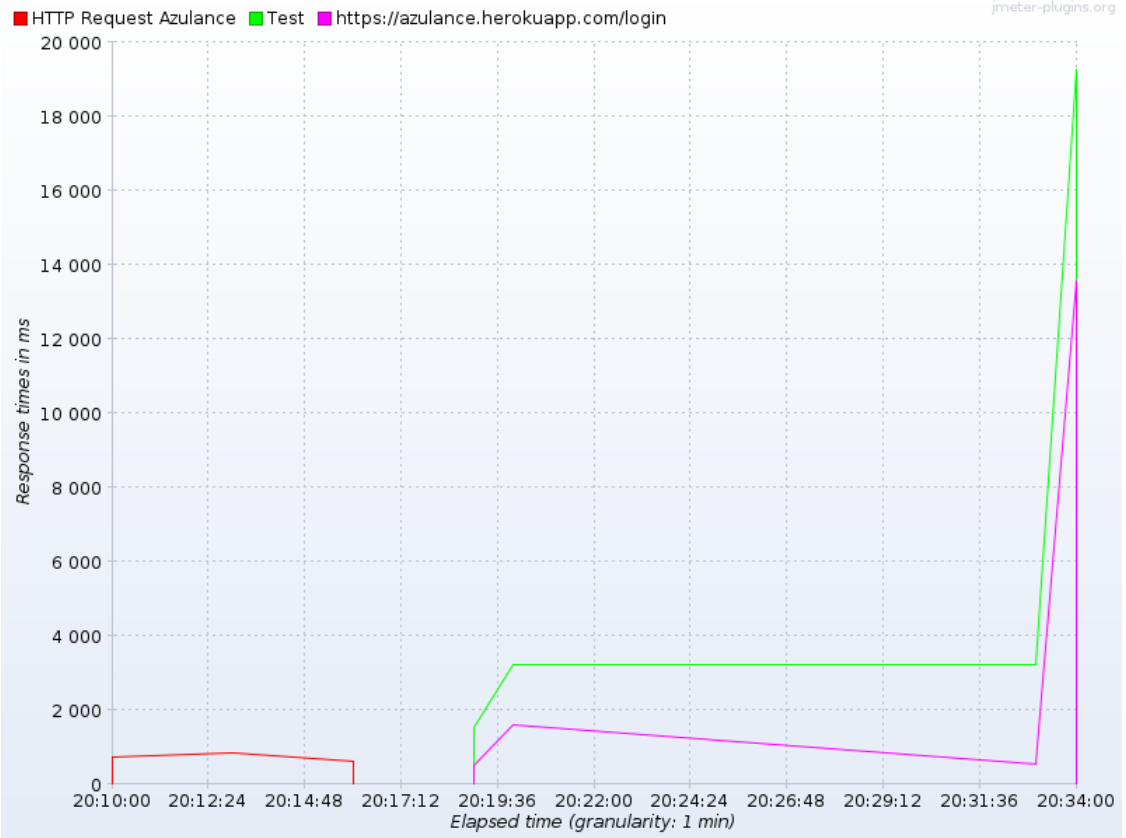
Pour générer nos graphiques, nous avons utilisé le plugin **Graphs Generator Listener**. Avec le Listener **View Results Tree**, nous générons un fichier CSV qui sera lu par Graphs Generator afin de pouvoir créer les graphiques attendus.



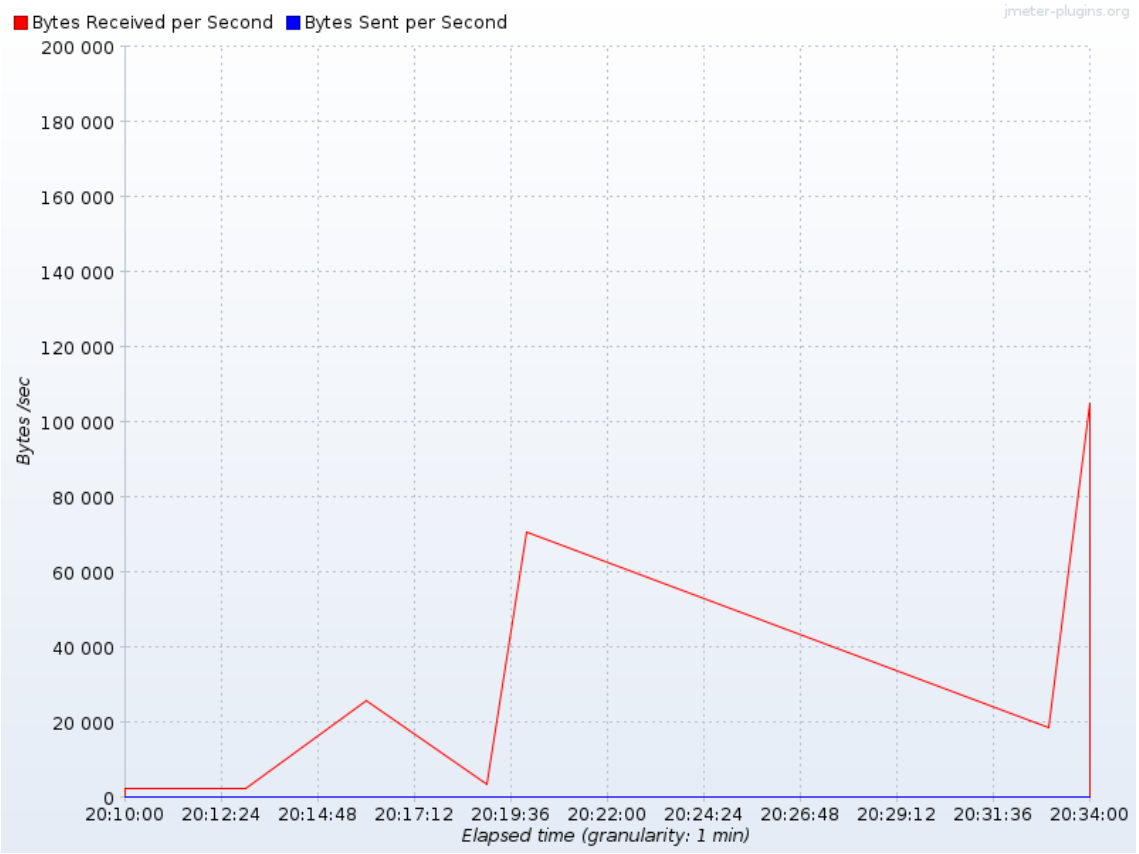
Graph Times VS Threads



Graph Throughput VS Threads



Graph Response times over time

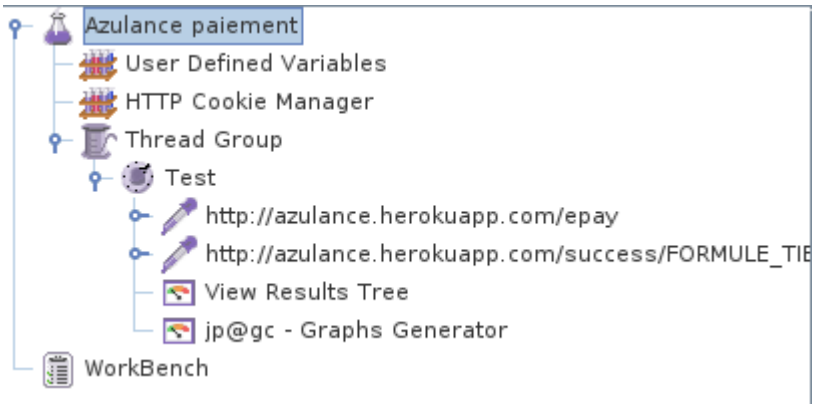


Graph Bytes Throughput over time

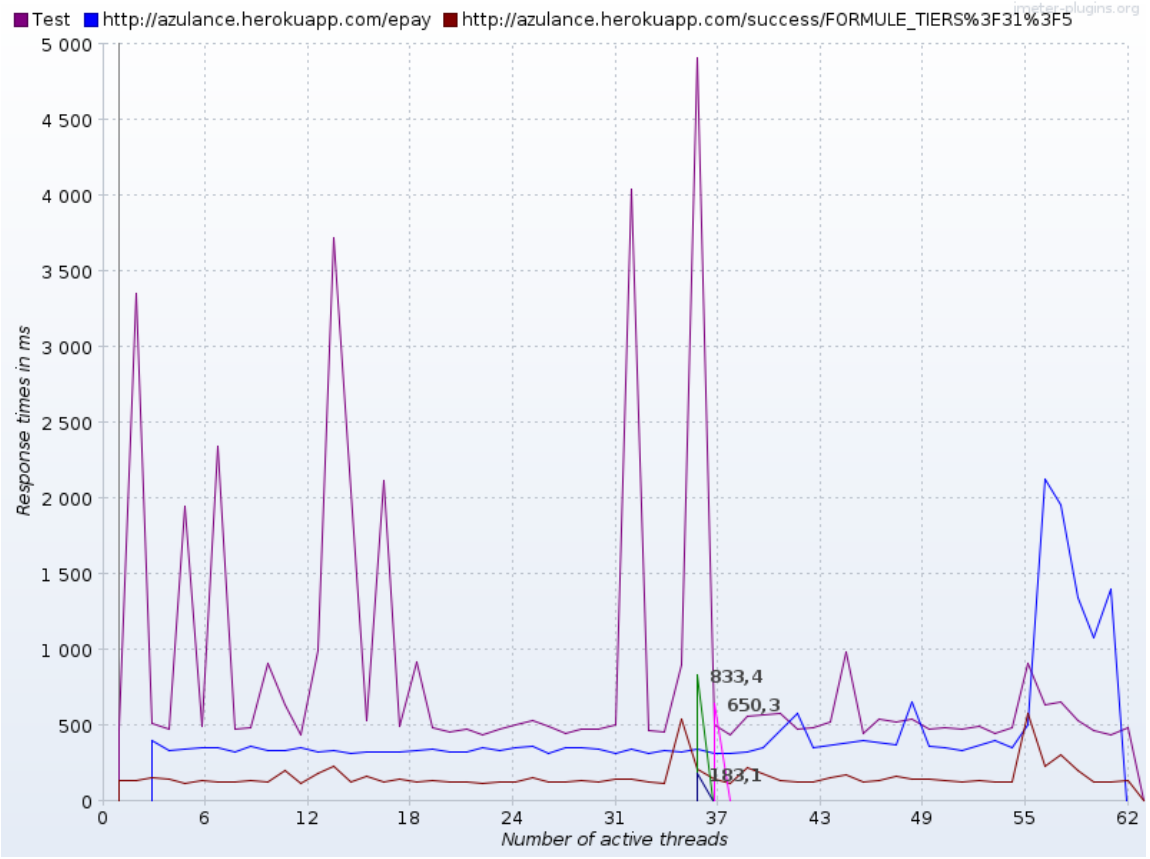
Nous voyons que le site tient la charge que nous lui avons imposée, malgré une lenteur parfois élevée. Il est cependant mieux d'héberger le site sur un serveur plus performant.

TEST 2 : Résultats de test réalisé sur le scénario de Paiement, en se basant sur le Cycle 2 Run 3.

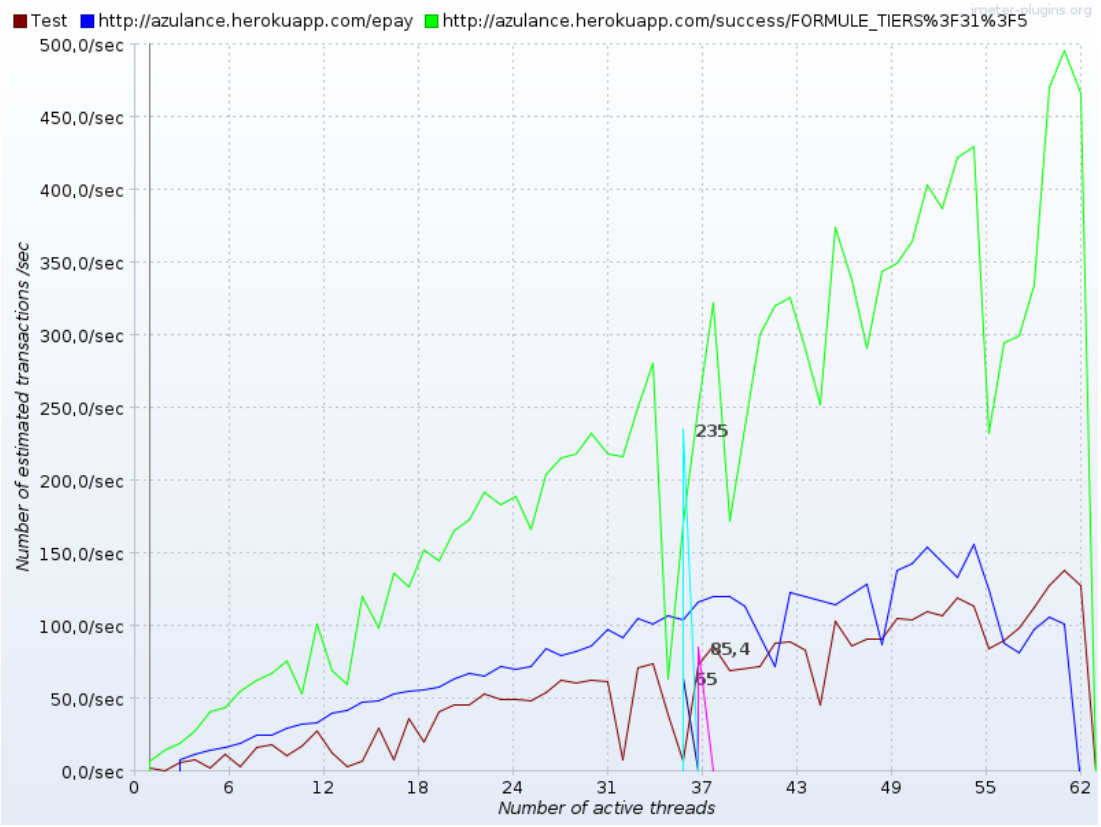
Nous avons d'abord généré un fichier .jmx à l'aide de BlazeMeter et avons gardé l'essentiel :



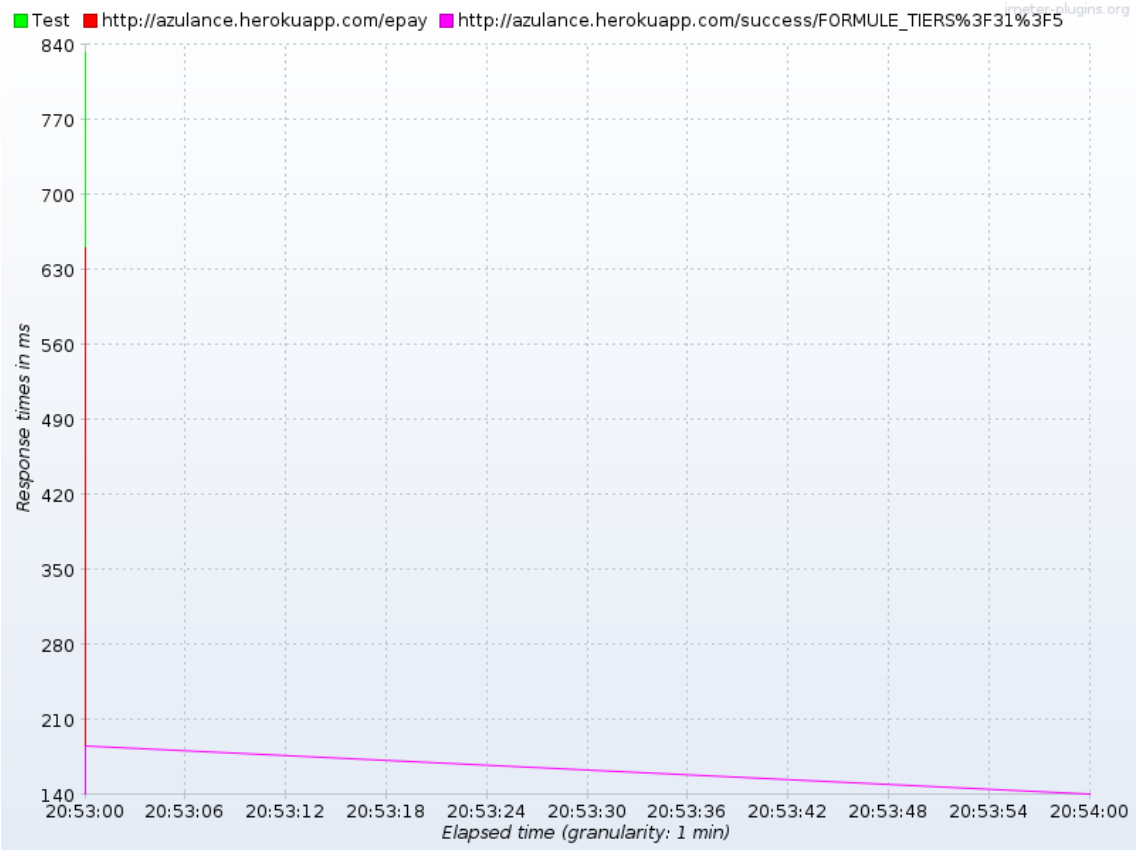
Pour générer nos graphiques, nous avons utilisé le plugin **Graphs Generator Listener**. Avec le Listener **View Results Tree**, nous générons un fichier CSV qui sera lu par Graphs Generator afin de pouvoir créer les graphiques attendus.



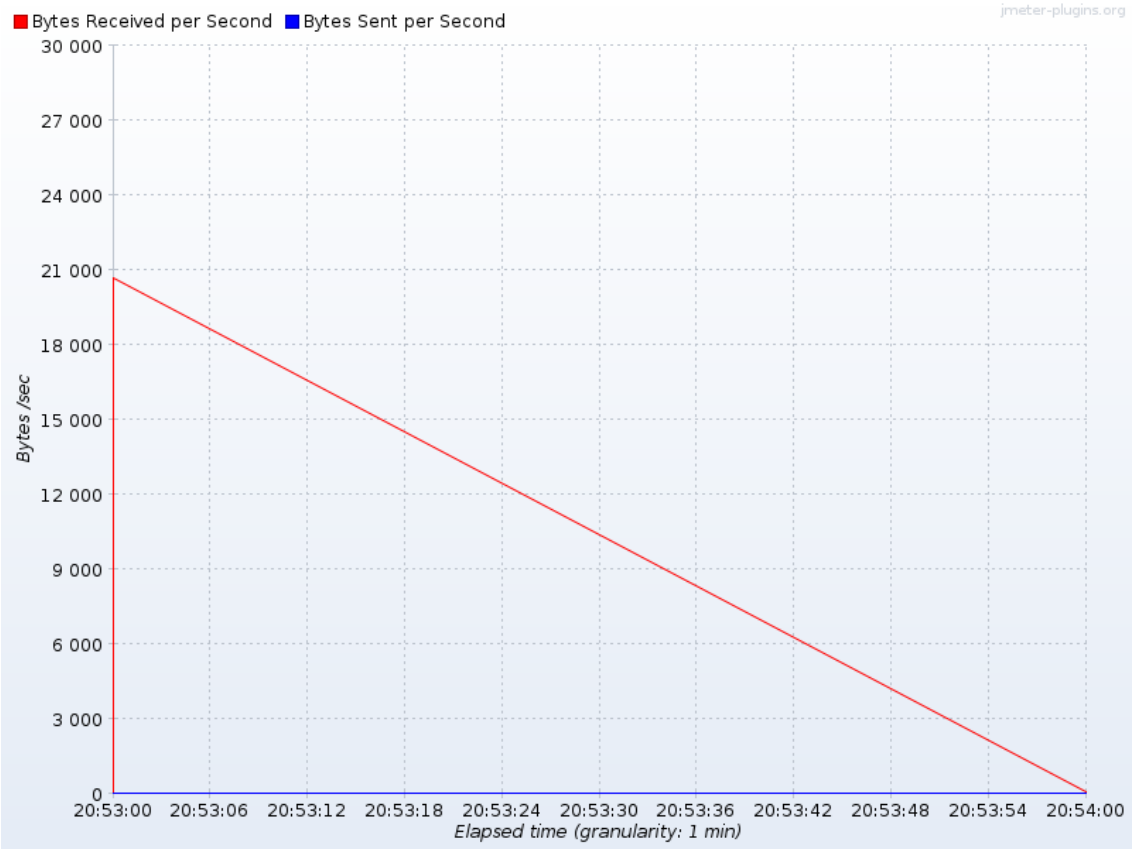
Graph Times VS Threads



Graph Throughput VS Threads



Graph Response Times over Time



Graph Bytes Throughput over Time

Nous constatons qu'il y a des irrégularités, parfois des pics de temps d'attente.