

Guillaume Noguera

Axel Bento Da Silva

Luc Gigon

JEE - PRojet

3ème année, dlm 2017 - 2018

**Qualité logiciel**

**Spécification et Tests**

**Kebab Service**

2018

1. Introduction

Dans le cadre du cours JEE, nous devions faire un projet de site web. Nous avons décidé de faire un site de commande et de vente de kebab. Nous l’avons appelé « Kebab Service ». Le site permet à des clients de commander des kebabs, et il permet au gérant de valider les commandes. En parallèle de la création du projet, nous avons dû mettre en pratique les techniques de qualité logicielle apprise dans le cours du même nom sur notre projet JEE.

Ce rapport montre les différents aspects de qualité logicielle appliquée à notre projet.

1. Stratégie de tests

|  |  |
| --- | --- |
| Caractéristique du produit | Ce produit doit être capable de servir à un kebabier dans la gestion des commandes des clients au quotidien.  Côté utilisateur, le but est de leur permettre de passer une commande en spécifiant les ingrédients composant leur kebab.  Côté kebabier, cela simplifie son travail car les commandes seront visible facilement et il pourra les valider au fur et à mesure. |
| Objectifs de test | Les éléments importants à tester sont les suivants :   * Les données à insérer dans les formulaires, * L’authentification au site, * La charge lors des moments de grande affluence (à midi et le soir), * La gestion des commandes. |
| Périmètre de test | Nous effectuerons ici uniquement les tests système (essentiellement performance et sécurité) ainsi que les tests de validation (utilisation et fonctionnalités cité dans les objectifs de test).  Nous partons du principe que si les fonctionnalités sont validées, il n’y a pas de raison ici de tester les niveaux unités et intégration |

1. Plan de tests
   1. Niveau de test
      1. Test système

Nous testons deux aspects de l’application web.

**La sécurité**: L’application web possède deux parties, la première à destination des clients qui viennent commander un kebab. Et l’autre partie réservée au kebabier qui lui permet de voir les commandes et les valider. Pour tester, nous créons deux utilisateurs, un client et un admin, le premier ayant les droits « user » et le second ayant les droits « admin ». Un troisième rôle existe, il s’agit du « cook », le cuisinier qui peut voir les commandes et les valider, mais pas les supprimer.

Règles pour les droits :

|  |  |  |
| --- | --- | --- |
|  | Accès | Droits |
| Admin | Tout le site | Lecture/Ecriture sur toute la BD |
| Client | Seulement la partie accueil et commande | Ecriture sur la BD |
| Cook (cuisinier) | Tout le site | Lecture/Ecriture partielle sur la BD |

**Les performances :** comme l’application web tourne sur un serveur local, il n’est pas réellement possible de tester les performances en matière de test de charge. Cependant, nous allons quand même les énumérer théoriquement dans ce rapport.

* + 1. Test de validation

**Les fonctionnalités :** plusieurs fonctionnalités essentielles seront testées selon le tableau suivant :

|  |  |
| --- | --- |
| Login | Connection d’un utilisateur en fonction de ses droit, pour autant qu’il ait entré un login et un mot de passe correspondant à ce qui est stocké dans notre BD. |
| La commande | Les informations dans le formulaire de commande doivent être correctes (type de kebab, ingrédient, sauce, heure) et stocker dans la BD. |
| Le panneau de contrôle | Voir un détail de chaque commande et permet de valider et/ou de supprimer les commandes réalisées. |

**L’ergonomie :** afin d’avoir un design acceptable, nous avons utilisé un Framework CSS s’appelant « Semantic UI» par-dessus les pages générer par JEE. Nous vérifierons donc que toutes les pages de l’application web utilisent bien le Framework CSS, nous testons notamment :

* Que le Responsive Design soit fonctionnel
  + Menu s’adapte selon la taille de l’écran
  + Largeur de la page adapté à la taille de l’écran
* Que tous les boutons respectent le Framework
* Que les formulaires respectent le Framework
  1. Infrastructure
     1. Outils de test

**Selenium IDE** : Extension sous le nom de Katalon Automation Recorder sur Chrome permettant d’enregistrer une suite d’actions qu’il sera possible de rejouer à volonté.

**Apache JMeter** : Outil permettant de tester les performances d’une application en simulant des utilisateurs ainsi que d’enregistrer les temps de réponse. Il permet également d’enregistrer automatiquement le scénario afin de gagner du temps et de permettre son utilisation à large public.

* + 1. Environnement de test

Tous les tests seront effectués depuis le PC personnel des testeurs. L’application étant hébergé en local (localhost), voici donc la configuration des serveurs :

* Luc Gigon : Wamp Server 3.1 sur un OS Windows 10 Professionnel

Chaque test sera effectué sur 2 navigateurs différents :

* Google Chrome v.65
* Firefox v.59.0

1. Spécification détaillées des tests
   1. Cluster

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | 1 |  | **ID** | 2 |
| **Cluster** | Authentification |  | **Cluster** | Commande |
| **Date** | 25.04.2018 |  | **Date** | 25.04.2018 |
| **Version** | 0.1 |  | **Version** | 0.1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | 3 |  | **ID** | 4 |
| **Cluster** | Panneau de contrôle |  | **Cluster** | Validité des liens |
| **Date** | 25.04.2018 |  | **Date** | 25.04.2018 |
| **Version** | 0.1 |  | **Version** | 0.1 |

* 1. Cas de tests

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Type** | **Description** | **Résultat attendu** | **Priorité** |
| 1.1 | Fonctionnel | L’utilisateur entre un login et un mdp présent et correspondant dans notre base de données. | Connexion réussie | 1 |
| 1.2 | Fonctionnel | L’utilisateur entre un login erroné | Connexion échouée | 1 |
| 1.3 | Fonctionnel | L’utilisateur entre un mdp erroné | Connexion échouée | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Type** | **Description** | **Priorité** |
| 2.1 | Fonctionnel | Un client peut commander un kebab. | 2 |
| 2.2 | Fonctionnel | Un client peut sélectionner le type de kebab (sandwich ou durum). | 2 |
| 2.3 | Fonctionnel | Un client peut sélectionner les ingrédients présents dans son kebab. | 2 |
| 2.4 | Fonctionnel | Un client peut sélectionner les sauces présentes dans son kebab. | 2 |
| 2.5 | Fonctionnel | Un client peut renseigner l’heure de la commande | 2 |

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Type** | **Description** | **Priorité** |
| 3.1 | Fonctionnel | La liste des commandes n’affiche pas les commandes supprimées. | 2 |
| 3.2 | Fonctionnel | La liste des commandes affiche les commandes passé mais non validées 🡪 « to do !» | 1 |
| 3.3 | Fonctionnel | La liste des commandes affiche les commandes validées mais pas encore supprimées 🡪 « destroy » | 1 |
| 3.4 | Fonctionnel | Le bouton « to do ! » permet de valider une commande. | 2 |
| 3.5 | Fonctionnel | Le bouton « destroy » supprime une commande de la liste. | 2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Type** | **Description** | **Résultat attendu** | **Priorité** |
| 4.1 | Fonctionnel | Tous les liens (barre de menu, footer, etc) doivent être fonctionnels | Aucune erreur de type 404 | 1 |

* 1. Tests de charge

Durée d’un scénario : 1 heures (env. 11h30 ~ 12h30, 19h00 ~ 20h00 et 15h00 ~ 16h00)

Un client effectue en moyenne 5 action par minute.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Scénario** | **Cas de test** | **Charge (user)** | **Débit (Tx/min)** | **Pacing Time** | **Ramp-up** |
| Midi | TC1 | 50 | 5 \* 50 = 250 | 60/5 = 12 | 10 min |
| Soir | TC2 | 40 | 5 \* 40 = 200 | 60/5 = 12 | 10 min |
| Journée | TC3 | 5 | 5 \* 5 = 25 | 60/5 = 12 | 10 min |

1. Rapport d’exécution des tests

Revu des tests fait jusqu’au 25.04.2018 :

Tests fait avec Katalon selenium sur Firefox (ceux-ci ont été fait directement sur le browser, pas d’implémentation de test JUnit) :

1. Tests d’authentification (1.1 / 1.2 / 1.3)
2. Tests sur les commandes (2.1 / 2.2 / 2.3 / 2.4 / 2.5)

Il n’a pas été possible de faire plus de tests. En cause, le développement s’est révélé être difficile pour plusieurs raisons :

* Problème de version de Glassfish et sa configuration,
* Problème avec la connexion base de données,
* Difficulté de « gitter » le projet et de le reprendre sans perdre des heures pour reconfigurer le tout.