ZiApi

Cahier de recette

Ceci est notre cahier de recettes- officiel du server web http/1.1

Développeurs :

* Armand Morgan
* Candan Caner
* Toumi Majdi

Table des matières

1. Introduction
2. Organisation
   1. Taches qui incombent au client.
   2. Taches qui incombent au fournisseur.
   3. Communication avec le client.
   4. Ordre de réalisation des tests.
      1. Test unitaire.
      2. Test de validation.
      3. Test d’intégration.
3. Etapes
   1. Recette d’usine
   2. Recette d’utilisateur
4. Anomalies
   1. Fiche d’anomalie
5. Annexes : séries de test.

Introduction

Notre cahier de recette regroupe tous les tests qui devront être effectués et validés par les deux parties (clients et fournisseur). Il spécifie un système de communication qui permet aux deux parties de communiquer pour s’échanger les rapports de tests effectués.

Chaque test est effectué dans un environnement bien défini permettant de bien appréhender les différents cas particuliers de fonctionnement du logiciel.

Organisation

1. **Taches qui incombent au client :**

Les parties qui vont suivre sont les fonctionnalités principales de notre serveur web. Il doit donc répondre aux définitions de celles-ci rappelé brièvement ici mais détaillée dans le cahier des charges du projet.

* Protocole http
* Système de modules
* Fichier de configuration XML
* Facilité d’installation
* …

Notre client se voit fournir une série de test bien définis pour valider ces fonctionnalités et si jamais cela ne l'était pas, il devra remplir un rapport et nous le communiquer (le mode de communication est expliqué plus loin).

1. **Taches qui incombent au fournisseur :**

Du côté du fournisseur, notre tâche principale est bien entendu la livraison du produit fini correspondant aux attentes du client. Parallèlement à ca nous fournissons les séries de tests que le client devra effectuer pour valider les spécifications attendues du logiciel.

Le moindre problème constaté par le client se devra de nous être renvoyé pour que nous puissions chercher la cause, la corriger et ainsi renvoyer au client une version fonctionnelle accompagnée de nouveaux tests pour valider la fonctionnalité défectueuse.

1. **Communication avec le client :**

A la livraison, le contact avec le client ne doit pas être interrompu, de part le fait que nous devons communiquer avec ce dernier pour qu’il puisse valider les fonctionnalités du logiciel ou faire état du problème le cas échéant.

Si un des tests échouait, le client doit nous envoyer un rapport d'anomalie avec les conditions et l'environnement utilisé pour faire ce test. Nous tenterons de reproduire le problème, le corriger puis de renvoyer une version fonctionnelle accompagnée d'un nouveau test validant la fonctionnalité défectueuse.

Si de nouveau un problème survenait sur ce même test, le client recommencerait son envoi de rapport d'anomalie et le processus se reproduira tant que le défaut subsistera.

1. **Ordre de réalisation des tests :**
2. **Test unitaire :**

Les différents tests unitaires sont présentés en annexes ; Elle contient des tests sur les unités suivantes :

Le serveur :

* Thread de ports.
* Thread de clients.
* Les clients.
* L’étape du parsing.

Les modules :

* Module SSL
* Module Perl
* Module PHP
* Module Ruby
* Module CGI BIN

1. **Test de validation :**

Les différents tests de validation nous permettent de vérifier les exigences du client ainsi que la performance et la robustesse de notre serveur notamment grâce a des benchmarking (teste a travers des outils externes, par exemple ab d’apache (Apache Benchmarking), etc.)

1. **Test d’intégration :**

Les différents tests d’intégration nous permettent de valider le fait que toutes les parties développées indépendamment fonctionnent bien ensemble.

N’importe quel module respectant l’API du projet, peuvent fonctionner dans le server.

Des modules de teste on été crée :

* mod\_ssl, connexion client-serveur sécurisée
* mod\_autoindex, gere le listing des dossiers et les index
* mod\_mime, listing des fichiers connus
* mod\_vhosts, gestion de plusieurs hosts sur des jails differents
* mod\_cgibin, execution des langages de scripting

Etapes

1. **Recette d’usine :**
2. **Protocole de test :**

A la fin de chaque étape du projet, une série de test est effectué :

* Les **tests unitaires** qui vont permettre de vérifier le bon fonctionnement d’un composant distinct du logiciel. Nous nous assurons que :
* Toutes les données correctement formatées sont traites correctement par notre serveur.
* Le traitement de données débouchées sur l’affichage d’un message d’erreur suffisamment explicite pour localiser les erreurs.
* Les gestions d’erreurs ne provoquent pas de comportement indéfinis conduisant a une interruption de service.
* Les **tests d’intégrité** permettant de vérifier que les composants se comportent de manière normale quand ils sont places dans leur contexte d’exécution normale. Voici notre démarche :
* Etablissement d’une configuration de base fonctionnelle minimale : notre serveur et éventuellement un ou plusieurs modules de base.
* Création d’une série de tests simple composes de requêtes sur des pages statiques et dynamiques, du contenu binaire (images, vidéos, document pdf, etc.)
* Rajout progressif de modules en complexifiant les tests afin de s’assurer du bon fonctionnement du serveur (des requêtes importantes à une fréquence de plus en plus élevée sont effectuées)

**Nous fournissons les tests.** Elles ont été réalisées à l’aide d’applications comme CppUnit, permettant d’effectuer des testes unitaires. (voir annexe Série de testes)

1. **Résultat et documentation fournis au client :**

Lors de la livraison de notre serveur web, nous fournissons également les documents suivants :

* La documentation de déploiement et configuration.
* La documentation de l’API nécessaire au développement de modules.
* Un fichier de configuration d’exemple.
* Une liste de tests a exécuté par le client pour valider le fonctionnement du produit.

1. **Recette d’utilisateur :**
2. **Recette fonctionnelle :**

Cette recette permet à l’utilisateur de voir rapidement si le logiciel fourni répond à ses besoins et est conforme aux fonctionnalités décrites dans le cahier des charges. Pour cela, nous procédons comme cela :

* Etablissement de sa propre liste de tests simples se complexifiant de façon croissante,
* Déploiement du logiciel dans un environnement proche de celui de production pour s'assurer qu'il répondra bien à ses attentes.
* Création d’une liste de ressources (fictives ou non) à utiliser lors des tests de fonctionnement.
* Vérification préalable du fonctionnement du logiciel (dans le cas où un défaut est découvert) afin de s'assurer que le logiciel n'a pas été altéré par une mauvaise configuration de celui-ci, auquel cas elle devra être revue.
* Remplissage d’une fiche d'anomalie et renvoie au fournisseur (si le défaut est bien réel).
* Recommencer la procédure de tests à son début (une fois le logiciel corrigé par le fournisseur,) pour s’assuré qu'aucune autre fonctionnalité n'a été affectée par la correction de l'anomalie

1. **Recette technique :**

Lors de la recette technique, le client établira sa propre liste de test permettant d’évaluer les performances du produit délivré par le fournisseur et ainsi vérifier si celui-ci sera utilise :

La recette technique sera conduite par le responsable informatique. Il devra s'assurer que l'application est bien compatible avec l'environnement technique dans lequel celle-ci sera utilisée. Il procédera comme suit :

* Le logiciel sera déployé, configuré et démarré pour vérifier si par défaut il ne produit pas d'erreur d'installation ou de configuration,
* Un test de bande passante pourra être effectué entre la machine où est déployé le serveur et celle utilisée pour exécuter les tests, ceci afin de vérifier que la liaison est suffisante pour des tests complexes de charge,
* On effectue une série de tests simples en vérifiant que le logiciel réponde correctement. On augmentera la charge des tests de façon critique pour essayer de produire une interruption de service,
* On journalise les tests effectués dans un rapport où l'on consigne la requête, le résultat, la conformité, le temps nécessaire à la réponse,
* On journalise aussi les erreurs rencontrées.
* On vérifie que le serveur fonctionne de manière stable sur une longue période d'activité.
* On vérifiera les résultats des tests avec ceux des tests usine et en cas d'important écart on recherchera si celui-ci ne vient pas d'un problème local et le cas échéant on avisera le fournisseur grâce à un rapport d'anomalie.

Anomalies

1. **Fiche d’anomalie :**

Si lors des tests fournis avec le logiciel, le client détecte un défaut il devra, comme mentionné plus haut, établir une fiche d'anomalie la plus claire possible et contenant au moins les informations suivantes :

* La version du logiciel ainsi que sa date de mise en circulation,
* Le fichier de configuration utilisé pour les tests,
* La liste des différents modules chargés par le logiciel ainsi que leur version,
* La description complète de l'environnement matériel et logiciel utilisé
* Le test ayant provoqué l'erreur ou :
  + Si c'est un problème technique :
    - La requête utilisée
    - La ressource ciblée par la requête
    - L'erreur renvoyée par le serveur
  + Si c'est une anomalie de fonctionnement :
    - La fonctionnalité testée
    - Le résultat attendu
    - Le résultat obtenu

**2) Anomalies rencontrées**

Tout au long du développement du projet, des anomalies ont été rencontrées.

Voici une liste :

* Fuite de mémoires
* Transfert de données pas optimisé
* …

Toute ses anomalies ont été corrige chez le fournisseur.

Annexes : séries de tests

Nos testes sont classées en catégorie :

* **Server :**