

J. Racaud;A. Simon;J. Harrault;J. Blondeel;S. Daguenet;F. Corradin

Music Sheet Writer

Music Sheet Writer

Stratégie de Tests

Objectifs du document

Résumé

Glossaire

***– R –***

Release : Mise à disposition d’une version de l’application. Elle peut être privée, semi publique ou publique. En général une release fait état de la version commerciale (ou plus détaillée pour des releases fermées).

Répartition des charges :

***– S –***

Symfony : Framework PHP pour créer des sites web et des applications web.

Script : Programme qui exécute plusieurs taches à la suite.

***– B –***

Bundle: Un bundle est une partie du projet qui peut être réutilisée par un ure projet.

***– F –***

Framework: Un Framework est un ensemble de composants qui permettent de structurer et de mettre en place les fondations d’un logiciel.

Description du document

|  |  |
| --- | --- |
| Titre | Music Sheet Writer : Stratégie de Tests |
| Date de création | 22/10/2015 |
| Date de publication | 15/11/2015 |
| Auteur | J. Racaud;A. Simon;J. Harrault;J. Blondeel;S. Daguenet;F. Corradin |
| Responsable | Jonathan Racaud |
| E-mail | musicsheetwriter\_2017@labeip.epitech.eu |
| Sujet | Stratégie de Tests |
| Version du modèle | 1.0 |

Tableau des révisions

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Auteur** | **Section(s)** | **Commentaire** |
| 22/10/2015 | Jeremy HARRAULT | Toutes | Création du document |

Sommaire

[1. Introduction 1](#_Toc435265369)

[1.1. Objectif de l’EIP 1](#_Toc435265370)

[1.2. Principe de base du système 1](#_Toc435265371)

[1.3. Documents de référence 1](#_Toc435265372)

[2. Version/release 2](#_Toc435265373)

[2.1. Site Web 2](#_Toc435265374)

[3. Types de tests et périmètre 3](#_Toc435265375)

[3.1. Tests Unitaires 3](#_Toc435265376)

[3.2. Tests fonctionnels 3](#_Toc435265377)

[3.3. Tests de performances 3](#_Toc435265378)

[4. Livrables 3](#_Toc435265379)

[4.1. Cas de tests 3](#_Toc435265380)

[4.2. Rapport d’exécution de tests 3](#_Toc435265381)

[4.3. Rapport sur le suivi global de la qualité 5](#_Toc435265382)

[5. Types de tests 5](#_Toc435265383)

[5.1. Tests Unitaires 5](#_Toc435265384)

[5.2. Tests Fonctionnels 6](#_Toc435265385)

[5.3. Tests de performances 7](#_Toc435265386)

[6. Estimation Globales de la charge de travail 7](#_Toc435265387)

[7. Outils 7](#_Toc435265388)

[7.1. Dépôt de tests 7](#_Toc435265389)

[7.2. Gestion de rejets/erreurs 8](#_Toc435265390)

[7.3. Autres outils 8](#_Toc435265391)

# Introduction

## Objectif de l’EIP

## Principe de base du système

## Documents de référence

# Version/release

## Site Web

Voici la liste des versions majeures de l’application Web. Leurs numéros de version seront notés sous la forme X.0 où X est le numéro de la release majeur. Ce nombre sera incrémenté de 1 à chaque release.

D’autres releases mineures pourront être faites si :

* Une fonctionnalité rapide à implémenter est fortement demandée par la communauté et/ou devient urgent à implémenter pour satisfaire l’expérience MusicSheetWriter.
* Des « bugs » sont trouvés sur l’application, soit par l’équipe de développement, soit pas la communauté.

Une version mineure sera notée sous la forme X.Y où X est le numéro de la version majeure en cours et Y est le numéro de la version mineure.

|  |  |  |
| --- | --- | --- |
| Date de la sortie | Numéro de version | fonctionnalité |
| 01/01/16 | 0.1 | API gestion de compte utilisateur   * Création de compte utilisateur * Connexion * Gestion des partitions favorites / partage * Gestion des abonnements |
| 04/03/16 | 0.2 | Gestion de la communauté   * Rechercher un utilisateur / partition * Consultation utilisateur / ajout de partition |
| 08/04/16 | 0.3 | Vitrine et Front end   * Vitrine du site web * Plateforme achats et téléchargements |
| 08/04/16 | 1.0 | Version finale |

# Types de tests et périmètre

## Tests Unitaires

Les tests unitaires seront effectués sur l’ensemble des fonctions de chaque classe de service pour permettre de vérifier le fonctionnement de tous les services de l’application. C’est test vont permettre directement de vérifier que les fonctions de bas niveau fonctionnent parfaitement en évaluant chaque valeurs de retour pour l’ensemble des cas possibles sur une opération.

## Tests fonctionnels

Les tests fonctionnels sont un ensemble de tests unitaire qui vont permettre de vérifier le bon déroulement d’une fonctionnalité de l’application comme par exemple l’inscription d’une personne sur le site internet. Ces tests permettent entre autre la validation d’une version et d’organiser les tests de telles manière qu’il soit plus compréhensible par le développeur.

## Tests de performances

Les tests de performances ont pour but de vérifier les temps de réponses sur des requêtes utilisateurs dans plusieurs conditions pour s’assurer de l’état du service.

# Livrables

## Cas de tests

### Site web

#### Tests Unitaires

Les tests unitaires seront exécutés par de simple fonction PHP qui vont vérifier la valeur de retour d’une fonction.

Chaque test unitaire sera regroupé par fonctionnalité dans une même méthode qui représentera le test pour une seule fonctionnalité.

#### Tests Fonctionnels

Le Framework Symfony est utilisé pour le développement du site web les différents cas de test fonctionnels seront regroupé par module, i.e. une classe de test pour la parties gestion des utilisateurs, une autre pour la gestion des partitions…

Chaque classe de test sera regroupée dans le bundle correspondant sous le répertoire /Test du bundle.

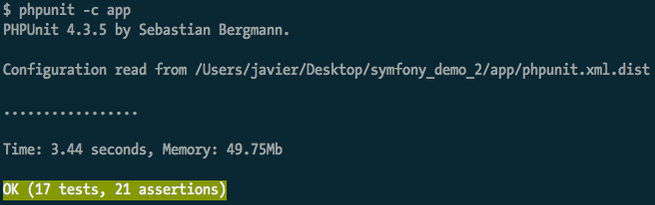
#### Tests de performances

Les tests de performances seront exécutés par l’outil Pylot ou chaque test sera d’écrit dans un fichier XML pour tester plusieurs requêtes vers le service web.

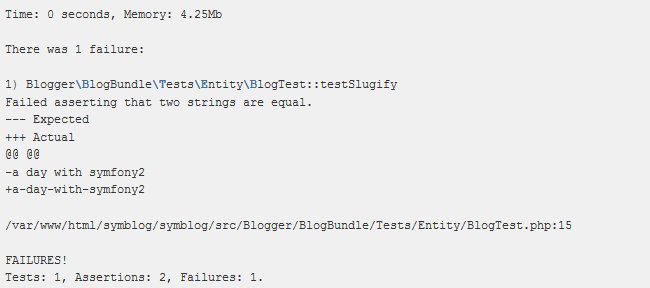
## Rapport d’exécution de tests

Tous les rapports d’exécution de tests seront fournis dans des fichiers au format texte pour les tests unitaire et fonctionnels et dans des graphes pour les tests de performances.

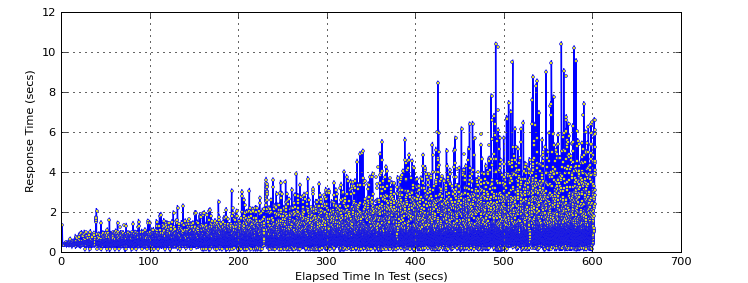
Dans le cas où il n’y a pas d’erreur le nombre de test passé s’affiche avec le message OK.



Si un test échoue un message d’échec apparait avec la ligne et la classe dans lesquels le test a échoué.



Les résultats des tests de performances seront sous la forme de graphes comme celui-ci :



## Rapport sur le suivi global de la qualité

Avant chaque mise à jour du projet sur le SVN il est impératif que le projet passe tous les tests de fonctionnalités cela veut dire qu’il ne doit y avoir aucun échec lors du lancement des commandes des tests sur les fonctionnalités.

# Types de tests

## Tests Unitaires

### Objectif

Le but de c’est tests va être de comparer la valeur de retour d’une fonction avec la valeur attendu.

Cela permet de retrouver facilement le moment ou le programme ne fonctionne pas comme prévue et de trouver le problème rapidement.

### Environnement et conditions de réalisation

Les tests unitaires pour le site web seront exécutés sur la partie interface utilisateur et sur l’API.

Avant le mise en production les tests seront effectuer premièrement un environnement de développement puis une fois déposé sur le SVN ils seront testés en environnement de production pour vérifier que le comportement soit bien le même.

En environnement de développement les serveurs de test ne seront pas configure dans des conditions optimales les serveurs n’ont pas besoin de configuration de sécurité ou de performance particulières la machine hébergeant l’application est moins performante. Contrairement à l’environnement de production.

### Configurations particulières

### Planning et charge

Les tests unitaires d’une partie du site devront être effectue avant les date limites de la réalisation de la partie prévue dans le gant.

### Critère de démarrage des tests

Pour démarrer les tests le serveur web, de base de données et de messagerie doivent être fonctionnels et bien configuré.

### Critères de passage/échec

Un test unitaires est considéré comme un échec du moment où les deux valeurs de comparaison sont différentes.

## Tests Fonctionnels

### Objectif

Le but des tests fonctionnels et de vérifier qu’une certaine fonctionnalité de l’application fonctionne parfaitement. Pour cela un test fonctionnel est composé de plusieurs tests unitaires qui vont vérifier l’ensemble des critères nécessaires pour juger qu’une fonctionnalité de l’application marche correctement.

### Environnement et conditions de réalisation

Avant le mise en production les tests seront effectuer premièrement un environnement de développement puis une fois déposé sur le SVN ils seront testés en environnement de production pour vérifier que le comportement soit bien le même.

### Configurations particulières

### Planning et charge

Les tests fonctionnels d’une partie du site devront être effectue avant les date limites de la réalisation de la partie prévue dans le gant.

### Critère de démarrage des tests

Pour démarrer les tests le serveur web, de base de données et de messagerie doivent être fonctionnels et bien configuré.

### Critères de passage/échec

Un test fonctionnel est considéré comme un échec lorsqu’au moins un des tests unitaires le composant a échoué.

Un test fonctionnel est considéré comme un échec lorsqu’au moins un des tests unitaires le composant a échoué

## Tests de performances

### Objectif

Les tests de performances ont pour but de vérifier les performances sur l’ensemble des services de l’application pour s’assurer d’un temps de réponse cohérent entre le serveur et l’utilisateur.

### Environnement et conditions de réalisation

Ces tests sont réalisés uniquement en environnement de production. Dans des conditions d’utilisation extrême pour pouvoir connaitre les limites de l’API et du site web sur les serveurs de productions.

### Configurations particulières

Les configurations pour les tests de performance seront définit dans des fichiers XML utilisé par Pylot pour d’écrire les tests.

### Critère de démarrage des tests

Pour démarrer les tests l’application doit être configurée pour être en mode de production.

### Critères de passage/échec

Les critères de passage à échec sont définit par plusieurs facteurs qui prennent en compte le temps de réponse, la complexité de la requête et la charge actuelle du serveur.

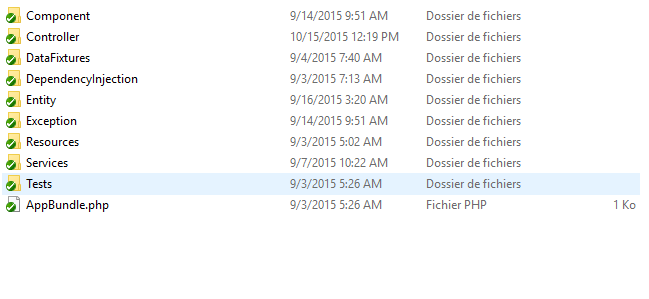
# Estimation Globales de la charge de travail

# Outils

## Dépôt de tests

Les dépôts de tests seront situés pour l’application web dans un répertoire tests localisé dans le répertoire du bundle spécifique de l’application.

Ex :



## Gestion de rejets/erreurs

La gestion des rejets de tests pour les scripts de fonctionnalité sera effectuée par les scripts de mise en production qui ne déploieront pas l’application tant que les tests ne seront pas valides.

## Autres outils

### Outils d’automatisation des tests

#### PHPUnit

Pour effectuer les tests sur le site web aussi bien sur la partir interface utilisateur que les tests en arrière-plan on utilise PHPunit qui lance tous les tests regroupés dans les classes des bundles du projet.

Cet outil permet d’effectuer plusieurs tests unitaires possibles sur des valeurs de retour de fonction.

Les tests seront automatiquement lancés par le script de mise en production.

#### Pylot est un outil

Pylot est un outil gratuit et open source pour tester les performances et l’évolutivité des services web.