Plan de test

Client : Consortium MedHead

Projet : Preuve de concept (l'allocation de lits d'hôpital pour les urgences)

Table des matières

Objet du document

Selon l'Institute of Electrical and Electronics Engineers (IEEE), un plan de test est un document qui décrit l'étendue, l'approche, les ressources et la planification des activités prévues. Sa rédaction s'effectue généralement juste après la phase d'analyse des exigences.

Il existe deux types de plan de test :

- Le plan de test maître qui a pour but de décrire la stratégie de test sur un projet particulier.
- Le plan de test de phase qui a pour but de décrire de manière détaillée les activités à mettre en œuvre pour chaque niveau de test.

Dans le cadre de ce projet et uniquement pour la partie du projet concernant les développements du la plateforme de streaming, ce document est donc le plan de test maître et le plan de test de phase. Le découpage du document respect le standard IEEE 829-2008.

Norme IEEE pour la documentation des tests de logiciels et de systèmes

> Les processus de test déterminent si les produits de développement d'une activité donnée sont conformes aux exigences de cette activité et si le système et/ou le logiciel répond à l'utilisation prévue et aux besoins des utilisateurs. Les tâches du processus de test sont spécifiées pour différents niveaux d'intégrité. Ces taches de processus déterminent l'étendue et la profondeur appropriées de la documentation de test. Les éléments de documentation pour chaque type de documentation de test peuvent ensuite être sélectionnés. La portée des tests englobe les systèmes basés sur des logiciels, les logiciels informatiques, le matériel et leurs interfaces. Cette norme s'applique aux systèmes basés sur logiciel en cours de développement, de maintenance ou de réutilisation (héritage, COTS, éléments non en développement). Le terme logiciel comprend également le micrologiciel, le microcode et la documentation. Les processus de test peuvent inclure l'inspection, l'analyse, la démonstration, la vérification et la validation des logiciels et des produits système basés sur des logiciels. (IEEE 829-2008 est remplacée par ISO/IEC/IEEE 29119-1-2013, ISO/IEC/IEEE 29119-2-2013, ISO/IEC/IEEE 29119-3-2013 et ISO/IEC/IEEE 29119-4- 2015.)

Historique du document

Date	Version	Commentaires
19 septembre 2022	1.0	Document en version finale et à améliorer au cours du projet.

Références

Les documents et livrables qui ont permis la rédaction du plan de test sont les suivants :

- Document d'architecture
- Ennoncé des travaux d'architecture
- Solution building block
- Données de référence sur les spécialités NHS
- Récaputilatif de réunion du comité d'architecture
- Hypothèse de développement
- Principes de l'architecture
- Feuille de route de l'architecture
- PoC du service d'urgences
- Hypothèses de validation de projet

Introduction

Ce document traite uniquement de la réalisation du service d'urgence voulu pour le projet MedHead. D'autres documents joint au prijet approndiront les notions fonctionnelles, techniques et opératives pour réalisation de l'application.

Ce plan de test fait suite à la preuve de concept (POC) du service d'urgence. Il constitue un guideline pour les tests déjà réaliser et ceux à réaliser pendant les développements majeurs qui suivront.

Ce document doit être mise à jour régulièrement.

Élément à tester

Les éléments pour le moment sont les suivantes :

ID	Elément	Nature des tests
1	POC Service d'urgences	• Test d'intégration continue • Test E2E

Fonctionnalités à tester

Cette partie décrit les caractéristiques fonctionnelles et non fonctionnelles qui doivent être testées. Il contient les références aux documents de spécification des exigences.

Caractéristiques fonctionnelles

Caractéristiques fonctionnelles : Service d'urgence

Titre		**Consulter la liste des urgences postés**
User story	*En tant que*	Utilisateur ou services tiers
	Il veut	Consulter la liste des urgences postés
	De sorte que	La supervision du service d'urgence soit réalisable, comme par exemple consulté : les hopitaux destionations, les localisations sources des urgences, les identifiants publiques des patients.
Critère d'acceptation	*Etant donnée que*	Les urgences sont traité de facon discrète par le système, un opérateur humain doit pouvoir avoir le supervision.
	Quand	L'utilisateur ou un services tiers appel le service à l'aide d'une requete REST
	Alors	Il receptionne en retour toute la liste des urgences.

Titre		**Poster une urgences**
User story	*En tant que*	Utilisateur ou services tiers
	Il veut	Poster une urgence
	De sorte que	Il sache ou déposé le patient en fonction de sa pathologie.
Critère d'acceptation	*Etant donnée que*	Le coeur du service d'urgence est répondre à des prérogatives d'acheminenement des patients vers des hopitaux où ils pouront être traité avec soins.
	Quand	Le service d'urgence recoit un requete d'urgence contenant la clé publique du patient, l'adresse d'origine de l'urgence et la pathologie associée.
	Alors	Le systeme renvoie l'urgence complété en y indiquant l'hopitale où déposé le patient qui pocède un lit disponible.

Titre		Lire les informations de destination d'une urgence
User story	*En tant que*	Utilisateur ou services tiers
	Il veut	Poster une urgence et lire des informations de destination
	De sorte que	La destination du patient soit le plus proche possible en fonction de sa pathologie et de son adresse d'origine.
Critère d'acceptation	*Etant donnée que*	Le transport de patient en état d'urgence doit se faire le plus vite et donc par le biais du plus court chemin.
	Quand	Une urgence a complété est réceptionné par le service d'urgence.
	Alors	L'hopitale possédant la capacité de soin et les lits disponibles le plus proche doit être retourné dans l'urgence complété.

Caractéristique technique

Caractéristique technique : service urgence

- Le service hopital est appelé par celui des urgences au travers d'un mock.
- Toutes les fonctions développés pour les besoins du projet sont couverte par un test.
- A chaque relivraison de code, les tests doivent être à jour et correspondre à ce qui est demandé.

Nom du composant	**Service urgence**				
Étape à exécuter	**Réponse du système**	**Résultat actuel**	**Précondition**	**Test java spring**	**Réussi/ Raté**
1-receptionner une urgence à compléter	1-urgence complété selon l'hoptial/spécialisé/lit disponible le plus proche	1-Réponse vide car le mockage ne fonctionne pas bien.	1-Mockito et configuration de test	Raté	non
2-réceptionner une urgence dont la spécialité est disponible dans tous les hoptiaux du parc	2-urgence complété selon l'hoptial/spécialisé/lit disponible le plus proche	2-Réponse vide car le mockage ne fonctionne pas bien.	2-Mockito et configuration de test	Raté	non

Caractéristique technique : service hopital

- L'annuaire des hopitaux doit être jour au travers d'un systeme de gestion global des

hopitaux.

- La données de référence sur les spécialités NHS par hopitaux doit être à jour conformément au point précédent.
- La disponibilité des lits doit être synchrone avec la capacité réel des hopitaux.
- Toutes les fonctions développés pour les besoins du projet sont couverte par un test.
- A chaque relivraison de code, les tests doivent être à jour et correspondre à ce qui est demandé.

Nom du composant	**Service hopital**				
Étape à exécuter	**Réponse du système**	**Résultat actuel**	**Précondition**	**Test java spring**	**Réussi/ Raté**
1-Récupérer tous les hopitaux	1-liste de tous les hopitaux et de leurs spécialités	1-liste de tous les hopitaux et de leurs spécialités	1-liste préchargé d'hopitaux / spécialités	Réussi	oui
2-Réceptionner un requete rest contenant spécialité et coordonnée origine	2-l'hopital le plus proche qui possède des lits de disponibles pour la spécialité correspondante	2-La vitesse du CDN est correcte	2-Api gateway active et liste préchargé d'hopitaux / spécialités	Réussi	oui
3-Réceptionner un requete rest avec une spécialité inconnu	3-retour spécialité inconnu	3-retour spécialité inconnu	3-Api gateway active et liste préchargé d'hopitaux / spécialités	Réussi	oui

Caractéristiques non-fonctionnelles

ID	Critère d'acceptation	Mesure de la conformité	Valeur
1	Le système d'intervention d'urgence en temps réel est adapté aux incidents	Test de monté en charge et scalabilité	0,5 % de pannes
2	Le système d'intervention d'urgence en temps réel gère la latence concernant la disponibilité des lits des hôpitaux du réseau	Tests de perfomance, de synchronisation des bases et d'appel distant	99 % fiabilité
3	Le système d'intervention d'urgence en temps réel offre des solutions lorsqu'il n'y a pas de lits d'hôpital disponibles pour la spécialisation requise	*A appronfondir*	
4	Le système d'intervention d'urgence en temps réel répond dans les 200 millisecondes à la demande de lits	Test de performance dans le système cible	< 200 ms
5	Le système d'intervention d'urgence peut être interfacé par d'autres systèmes	Usage d'API Rest et de framework de sécurité	0 % fuite de données et piratage

Approche stratégique

Test d'integration continue

L'intégration continue est une méthode de développement de logiciel DevOps avec laquelle les développeurs intègrent régulièrement leurs modifications de code à un référentiel centralisé, suite à quoi des opérations de création et de test sont automatiquement menées. L'intégration continue désigne souvent l'étape de création ou d'intégration du processus de publication de logiciel, et implique un aspect automatisé (un service d'IC ou de création) et un aspect culturel (apprendre à intégrer fréquemment). Les principaux objectifs de l'intégration continue sont de trouver et de corriger plus rapidement les bogues, d'améliorer la qualité des logiciels et de réduire le temps nécessaire pour valider et publier de nouvelles mises à jour de logiciels.

Test BDD

La programmation pilotée par le comportement est une méthode de programmation agile qui encourage la collaboration entre les développeurs, les ingénieurs qualité et les intervenants non techniques ou commerciaux participant à un projet logiciel. Il encourage les équipes à utiliser la conversation et les exemples concrets pour formaliser une compréhension commune de la façon dont l'application doit se comporter. Le BDD combine les techniques et principes du développement piloté par les tests avec les principes de la conception pilotée par le domaine et de la conception orientée objet pour partager une méthode et des outils communs entre les équipes de développement et les autres parties prenantes.

Le BDD est largement facilité par l'utilisation d'un langage dédié simple utilisant des constructions en langage naturel qui peuvent exprimer le comportement et les résultats attendus. Cela permet aux développeurs de se concentrer sur les raisons pour lesquelles le code doit être créé, plutôt que les détails techniques, et minimise la traduction entre le langage technique dans lequel le code est écrit et le domaine de la langue parlée par les entreprises, les utilisateurs, les intervenants, la gestion de projet…

Test E2E

tests de bout en bout

Les tests de bout en bout sont une technique utilisée pour vérifier si une application comporte comme prévu du début à la fin. Le testeur doit se mettre dans la peau d'un utilisateur et dérouler les tests comme s'il utilisait véritablement l'outil mis à sa disposition. Cette technique permet de vérifier les intégration du back-office et autres webservices.

Les tests de bout en bout sont généralement exécutés après les tests fonctionnels et après avoir vérifié le fonctionnement du système. Il sont, autrement dit, réalisés juste avant la mise en production du site Internet ou de l'application mobile.

Critères de succès et d'échec

On considère pour cette itération du POC que la campagne de test est réussite si 100 % des tests sont à l'état réussi.

Certains tests ne sont pour le moment pas fonctionnel. Un court délais supplémentaire devrait permettre de résoudre les problématiques de mockage du service hopitale dans celui des urgences.

Environnement et outils

Environnement

ID	Nom	Définition	Etat
DEV	**Développement**	Machine local des développeurs	Disponible
CI	**Intégration continu**	Un pipeline d'intégration pointe sur cette environnement pour réaliser les tests et les déploiements	A réaliser
CERT	**Certification et homologation**	Environnement simulant une architecture finale. Cet environnement sera utile pour les tests de qualification et d'homologation.	A planifier
PROD	**Production**	Environnement finale du service d'urgence.	A planifier

Outils

Outil	Description
Serveurs	Serveurs hébergent les environnements utiles pour les développements et test de la plateforme de streaming : • Développement de tests • Développement logiciel • Intégration continue • Certification et homologation • Déploiement en prod
Ordinateurs	-
Test BDD	Les outils comme Cucumber, offre la possibilité de mettre en forme les spécifications des tests. L'outil de test BDD sera utile pour l'implémentation des scénarios. Cucumber offre aussi beaucoup de flexibilité à l'usage avec la plupart des langages de développement.

Outil	Description
Pipelines CI/CD	Les outils comme GitLab, sont utiles pour l'exécution automatisée des tests unitaires, fonctionnels et des tests d'intégrations. Un pipeline d'intégration et de déploiement continus (CI/CD) est une série d'étapes qui doivent être effectuées afin de fournir une nouvelle version d'un logiciel. Les pipelines CI/CD sont une pratique axée sur l'amélioration de la livraison de logiciels tout au long du cycle de vie du développement logiciel via l'automatisation. → Jenkins
Suivi des bugs et des incidents	Les outils de ticketing, par exemple Jira Software, permet le suivi et la résolution des incidents. Ce type d'outil facilite la communication inter-équipe et les analyses de bugs.
Reporting / Analyse qualité	Les outils comme Sonar Qube permet l'analyse de statiques et fourni des métriques sur la qualité du code, la couverture de code par les tests, les bugs – failles potentielles. Ces outils sont généralement personnalisables et peuvent être intégré au pipeline CI/CD afin d'automatiser la mise à jour des rapports à chaque déploiement.

Rôles et répartition des tâches

Liste des taches

Nom de la tache	Définition
Cycle de vie du logiciel	• Relatif au suivi du cycle de vie d'un logiciel
Qualification logicielle	• Relatif à la qualification des tests informatiques
Préparation des tests	• Relatif à la préparation des tests
Exécution des tests	• Relatif à l'exécution des tests informatiques
Collecte des résultats	• Rapports générés dans le cadre des campagnes de tests automatisés. • Rapports fournis par l'outil de suivi des campagnes de test manuels. • Rapports automatiques générés à l'aide des tableaux de suivi des bugs. • Consolidation des rapports • Transmission du rapport consolidé pour analyse des écarts avec l'architecture et les exigences.
Identifier les exigences du logiciel	• Relatif à la spécification des exigences logiciels
Identifier les exigences	• Le choix de ces outils dépend du type de test à effectuer.

Nom de la tache	Définition
le matériel de référence	• Différents outils sont recommandés pour différentes plateformes. • Les outils pour les tests d'applications garantissent la performance, la facilité d'utilisation et la fonctionnalité des applications sur une variété de dispositifs.
Identifier les exigences de l'utilisateur final	• Relatifs aux bonnes pratiques pour rédiger les exigences des projets logiciels
Préparer prépare un plan d'essai	• Spécifie les composants qui seront testés, les différentes activités de testing, les ressources requises et un planning des activités. Le Software testé définit les spécifications du test et les spécifications de la procédure du test, en documentant la méthode utilisée, le choix des différents essais et la séquence d'actions prévues pour leur exécution.
Identifier les activités de débogage et correction des erreurs	• Relatifs au test liés au changemet et au test de non régression • Méthodologie de débogage • Recherche d'outil permettant le débogage et la surveillance des applications (exemple dans ce lien).

RACI du plan de test

A définir en fonction d'une liste exaustive de parti prenant.

Besoins en formation, recrutement ou sous-traitance

A définir celon les profils selectionnés pour les développements, tests et infrastructure.

Planning des activités

Définit la documentation existante et fourni pour ce projet (cf. Roadmap projet).

Livrables

Les livrables de test fournis avant chaque itération :

- Stratégie de test pour les itérations de développement suivantes reprenant le modèle de ce document.

Les livrables de test fournis pendant les tests :

- Liste des scénarios éligibles à l'automatisation
- Scripts de test
- Données de test utilisées en entrée

- Matrice de traçabilité des exigences
- Journaux d'erreurs et d'exécution.

Les livrables de test sont fournis une fois les itérations terminées :

- Rapports d'incidents des tests
- Rapport de synthèse des tests
- Directives relatives aux procédures d'installation/de test
- Matrice de traçabilité des exigences
- Métriques et mesure de test

Approbation

Nom / Rôle	Date	Signature
Rudy Hoarau *Architecte Logiciel*		