Concept de test

Love Mirroring

|  |  |
| --- | --- |
| Classification | non classifié |
| État | en cours d'élaboration |
| Nom du programme | Love Mirroring |
| N° du projet | 1 |
| Chef de projet | Tim Allemann |
| Version | 0.1 |
| Date | 07.04.2020 |
| Mandant | Stéphane Gerber |
| Auteur/Auteurs | Tim Allemann, Sébastien Berger, Hans Morsch |
| Distribution |  |

Suivi des modifications

| Version | Date | Modification | Auteur |
| --- | --- | --- | --- |
| 0.1 | 26.03.2020 | Création de la documentation et première modification | Sébastien Berger |
| 0.2 | 28.03.2020 | Ajouts de tests | Tim Allemann |
| 0.3 | 02.04.2020 | Remplissage du dossier | Sébastien Berger |
|  |  |  |  |

Tableau 1: Contrôle des modifications

Description

Le concept de test décrit les objectifs des tests, les objets à tester, les types de test, l’infrastructure et l’organisation de test. Il englobe également la planification des tests et la description des cas de test. Une description détaillée est établie pour chaque cas de test. Elle représente la spécification du test. La planification des tests détermine le déroulement logique et temporel des tests. Le concept de test constitue la base sur laquelle l’organisation et l’infrastructure de test sont mises à disposition et les tests sont exécutés. Il est régulièrement actualisé si de nouvelles constatations sont faites.

# Objectifs des tests

Objectifs mesurables des tests applicables à tous les cas de test:

| No | Description | Grandeur de mesure | Priorité\* |
| --- | --- | --- | --- |
| 1 | Test de création de compte | Création du compte réussie et visible dans la base de données.  Profil établi avec succès via un quizz | N |
| 2 | Test de l’authentification | Authentification réussi, l’utilisateur peut accéder à l’application | N |
| 3 | Test de la gestion du compte | L’utilisateur peut faire du CRUD sur son compte | N |
| 4 | Test de visualisation des profils | L’utilisateur peut voir des profils qui lui correspondent (selon réponse au quizz) | N |
| 5 | Tester la gestion des rôles | L’administrateur peut faire du CRUD sur les rôles | N |
| 6 | Test l’outil de communication | L’utilisateur peut envoyer des messages dans les différents canaux | N |
| 7 | Test de l’abonnement | L’abonnement fourni les prestations proposées et le paiement est fonctionnel. | N |
| 8 | Test de la vérification par mail | L’email de l’utilisateur est vérifié | N |
| 9 | Test de la vérification par sms | Le téléphone de l’utilisateur est vérifié | N |
| 10 | Test de réinitialisation de mot passe | Le mot de passe est modifié | N |
| 11 | Test de la messagerie privée | Le destinataire doit recevoir le message ainsi qu’un mail si spécifié | 1 |
| 12 | Test du chat général | Le message doit s’afficher sur le chat général pour tous les utilisateurs | 1 |
| 13 | Test de l’algorithme de match | L’algorithme doit faire correspondre des profils à 70% | N |
| 14 | Test de l’API sécurisé | L’API ne doit être utilisé que par les clients et utilisateurs concerné | N |
| 15 | Test de l’intégration des langues | La langue affichée s’appuie sur la langue du navigateur | 3 |
| 16 | Test de l’intégration de la newsletter | La newsletter s’envoie chaque jour | 2 |
| 17 | Test du support client | Vérification que l’administrateur reçoit le mail par le biais du formulaire | N |
| 18 | Test du bot | Le bot doit filtrer les messages indésirables | 1 |
| 19 | Test du paiement | Le paiement doit être exécuté chez le fournisseur et être historisé sur notre application | N |
| 20 | Test de la vérification des comptes | Le compte doit être validé suite à la vérificaiton système | N |
| \* Priorité: N = nécessaire / 1 = haute, 2 = moyenne, 3 = basse | | | |

Tableau 2: Objectifs généraux des tests

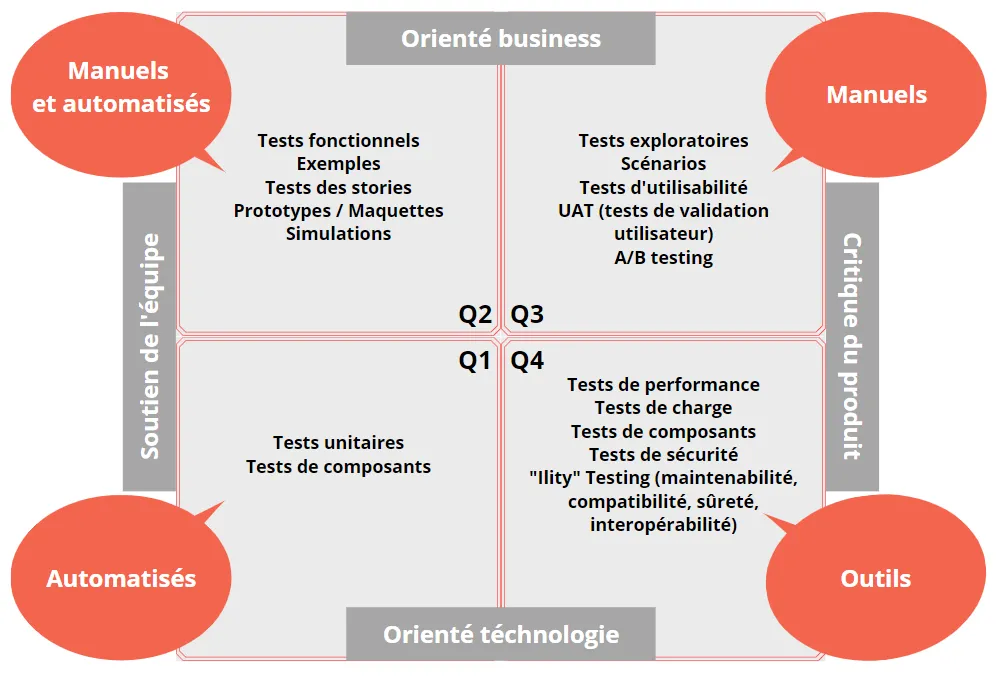
# Stratégie en matière de tests et niveaux de tests

Pour une procédure de test efficace, il est nécessaire d’élaborer une stratégie de test réfléchie (et éventuellement applicable dans un environnement agile):

Notre but est de fournir une application qui répondent aux exigences de notre mandant en donnant des points d’états durant l’avancée du projet et en évaluant la fiabilité de notre application grâce à nos tests.

Notre mode de fonctionnement serait de développer le test de la fonctionnalité avant de penser à sa réalisation. De plus nos user stories ne seront pas validées tant que les tests ne seront pas validés.

Nous souhaitons pour cela utiliser une matrice qui correspondrait à notre mode de fonctionnement :



* Quelles étapes de test existe-t-il? / À quelle étape de test faut-il commencer ?

1. Identifier les besoins
2. Focaliser sur les parcours stratégiques
3. Imaginer les comportements consommateurs les plus insolites
4. Séquencer et identifier les actions
5. Être précis sur les attendus

Pour éviter de faire des tests qui n’apportent pas une plus-value au projet nous allons procéder de la manière suivante :

1. Effectuer les tests unitaires dès que le code est rédigé pour voir si le fonctionnement de la méthode est ok.
2. Test de composants : faire un test de l’API ainsi que des différents services que nous utilisons dans notre projet
3. Tests des stories pour s’assurer que chaque livrable opérationnel
4. Tests fonctionnels afin de pouvoir avoir une vue d’ensemble de ce qui fonctionne et ce qu’il reste à faire. Cela nous permettrait d’avoir une vue d’ensemble du fonctionnement global de l’application.
5. Test de scénario : faire des tests en incarnant des types d’utilisateurs ou demander à des externes d’effectuer un test sur notre application. Cela nous permettrait d’avoir un retour externe.
6. Test de charge et performance : Pour voir si notre application tiendra le coup même en cas de surcharge. Cela s’appliquera notamment pour les canaux de communication pour voir si notre serveur tiendrait la charge en cas d’envoi de données en masse.

Pour pouvoir mener à bien nos tests il faudra respecter l’ordre qui a été mentionné plus haut afin de ne pas générer de retard et d’erreur involontaire et de se perdre dans les divers bugs qui pourraient survenir.

Afin de garantir que notre environnement de test a été efficace nous allons planifier un test d’utilisation réel par des utilisateurs (peut-être la classe) comme mentionné plus haut.

# Objets à tester

| No | Objet | Description |
| --- | --- | --- |
| 1 | Créer un compte | Il doit pouvoir créer un compte depuis la vue correspondante. |
| 2 | Authentification | Il doit pouvoir s’authentifier avec son compte. |
| 3 | Système de paiement | Il doit pouvoir choisir une formule d’adhésion et effectuer un paiement mensuel jusqu’au choix de l’arrêter |
| 4 | Gestion du compte | Il doit pouvoir gérer son compte et faire du CRUD |
| 5 | Outil de communication | L’utilisateur doit pouvoir communiquer dans différents canaux de communication. |
| 6 | Bot | Le bot doit pouvoir filtrer des messages qui ne respectent pas la charte d’utilisation de l’application. |
| 7 | Support client | L’utilisateur doit pouvoir contacter le support client pour soumettre différentes demandes. |
| 8 | Newsletter | L’utilisateur doit pouvoir s’abonner à la newsletter afin de garder un contact avec l’application et d’être averti des derniers profils qu’il peut visualiser |
| 9 | Vérification de compte | L’utilisateur doit passer par un processus de vérification multiples pour pouvoir valider son compte |
| 10 | Algorithme de match | Le système doit pouvoir déterminer en fonction de plusieurs paramètres |
| 11 | Gestion des rôles | L’administrateur doit pouvoir changer des rôles sur des utilisateurs |
| 12 | Traçabilité des connexions | L’administrateur doit pouvoir voir les informations sur les utilisateurs (connexions) |
| 13 | API Sécurisée | L’API doit être sécurisées et on ne doit pas pouvoir accéder aux controllers sans être authentifié |
| 14 | Compte externes | L’utilisateur doit pouvoir se connecter avec un compte externe |

Tableau 3: Objets à tester

# Types de test

| No | Type de test | Description |
| --- | --- | --- |
| 1 | Tests unitaires | Tester des bouts de code ainsi que leurs dépendances. |
| 2 | Test d’intégration | Vérifier que tous les codes de nos tests unitaires sont fonctionnels et collaborent bien ensemble si nécessaire. |
| 3 | Test système | Tester notre application en entier en appliquant des processus. |

Tableau 4: Types de test

# Couverture du test

## Aperçu des cas de test

| No | Objet à tester | Cas de test |
| --- | --- | --- |
| 1 |  |  |
| 2 |  |  |

Tableau 5: Couverture du test

## Évaluation des objectifs et couverture du test

# Cadre de test

## Conditions du test

Notre équipe seront les testeurs de l’application en termes de fonctionnalité, pour le test manuel nous allons proposer à des externes de tester l’application pour avoir un avis client et pas une vue développeur qui connait en soit l’application.

Notre équipe a déjà expérimenté des tests sur d’autres projets et elle pourra mettre ses compétences en œuvre pour rendre l’application livrable sans bug majeur.

Nous avons pensé demander à Monsieur Braillard et Madame Thirunavukarasu pour tester notre application.

## Classification des défauts

Les défauts constatés et les exigences non satisfaites (attentes) sont classés dans des catégories allant de 0 à 4. La catégorie 0 est uniquement utilisée pour présenter séparément un résultat irréprochable :

| No | Catégories de défaut | Description |
| --- | --- | --- |
| 0 | Sans erreur | Irréprochable et conforme aux exigences |
| 1 | Défaut insignifiant | Utilisation possible et utilité avérée; aucun défaut ne devrait survenir |
| 2 | Défaut léger | Utilisation possible; l’utilité n’est que légèrement réduite |
| 3 | Défaut grave | Utilisation encore possible; l’utilité est très limitée |
| 4 | Défaut critique | Inutilisable;  les fonctionnalités principales ne sont pas assurées;  impossibilité d’assumer la responsabilité de la mise en service (p. ex. pour des raisons de sécurité) |

Tableau 6: Catégories de défauts

La classification indique la gravité des conséquences et la charge de travail nécessaire pour corriger les défauts observés. Elle permet également d’obtenir un ordre de priorité approximatif pour l’élimination des défauts.

Si une catégorie de défauts entre 1 et 3 est atteinte, la réception du système/produit peut être effectuée sous réserve. Des mesures doivent toutefois être définies pour la suppression des défauts. Un contrôle complémentaire est impératif.

En revanche, si des défauts de la catégorie 4 sont constatés, la réception du système/produit ne peut être effectuée et le mandataire doit immédiatement prendre des mesures pour supprimer ces défauts. Il doit en outre initier une nouvelle réception.

## Conditions de début et d’arrêt

Le test doit débuter une fois que la fonctionnalité a été mise en place, il est intéressant d’appliquer la méthode Test-Driven Development. Donc le test débutera directement au début du développement de la fonctionnalité et s’arrêtera une fois que le test a été conçu.

Cela nous permettra d’effectuer des tests automatiques et de diriger notre développement plus facilement grâce au test.

# Environnement de test

Pour pouvoir tester notre interface nous ferons des tests manuels pour surtout identifier si l’application de ne comporte pas de bug graphique ou que les données ne sont pas correctement exploitées par les vues.

Pour la partie métier (controller et modèle) nous allons effectuer des tests automatiques avec le principe de test unitaire et tout ce qui s’y rapporte.

Cela nous permettra d’avance un peu plus rapidement dans notre développement sans devoir intervenir avec l’interface graphique à chaque fois et de faire la totalité des étapes à la main.

# Infrastructure de test

## Système de test

Nous allons utiliser les tests unitaires en. NETCore pour pouvoir définir nos tests

## Données de test

Les tests seront répertoriés sur un fichier Excel interne ou tout le monde pourra poster ses résultats.

Pour les tests globaux de l’application par des utilisateurs externes, nous allons leur mettre à disposition un fichier Excel leur permettant de décrire ce qu’ils voient durant leur expérience et si les tâches qu’ils devaient effectuer

## Utilitaires de test

Nous avons créé projet MSTest (.NETCore) ou nous écrirons nos tests. Le but est de pouvoir lancer des tests au fur et à mesure de notre développement pour avoir un bon aperçu de nos fonctionnalités.

A chaque test nous rédigerons un procès-verbal de test fournit par la méthode Hermès afin d’avoir une traçabilité de ce que l’on effectue sur nos applications.

En cas de non-réussite d’une fonctionnalité nous pourrons remonter dans les tests pour identifie à quel moment cela fonctionnait sans trop de problèmes.

# Organisation des tests

Voir dans la stratégie de test

# Description des cas de test

| ID / Désignation | T-001 | A1- Le test est réussi quand l’utilisateur reçoit un mail de confirmation de son inscription. | Nn p. ex. les exigences liées à la qualité dans le résultat Exigences concernant le système |
| --- | --- | --- | --- |
| Description | L’utilisateur doit pouvoir créer un compte depuis la vue correspondante. | | |
| Condition préalable au test | Les fonctionnalités d’inscription doivent avoir été remplie auparavant. | | |
| Étapes du test | Etape 1 : créer un utilisateur pour pouvoir l’envoyer au contrôleur  Etape 2 : envoyer l’utilisateur au contrôleur  Etape 3 : La méthode qui insert l’utilisateur dans la base de données | | |
| Résultat attendu | L’utilisateur doit avoir été insérer dans la base de données | | |

| ID / Désignation | T-002 | A2 – Le test est réussi quand l’utilisateur a pu correctement s’identifier sur l’application. | Nn p. ex. les exigences liées à la qualité dans le résultat Exigences concernant le système |
| --- | --- | --- | --- |
| Description | Il doit pouvoir s’authentifier avec son compte. | | |
| Condition préalable au test | Les fonctionnalités de connexion doivent avoir été définies auparavant et un utilisateur doit avoir été créé | | |
| Étapes du test |  | | |
| Résultat attendu |  | | |

| ID / Désignation | T-001 | Référence à l’exigence et critère d’acceptation | Nn p. ex. les exigences liées à la qualité dans le résultat Exigences concernant le système |
| --- | --- | --- | --- |
| Description | L’utilisateur doit pouvoir créer un compte depuis la vue correspondante. | | |
| Condition préalable au test | Les fonctionnalités d’inscription doivent avoir été remplie auparavant. | | |
| Étapes du test | Etape 1 : créer un utilisateur pour pouvoir l’envoyer au contrôleur  Etape 2 : envoyer l’utilisateur au contrôleur  Etape 3 : La méthode qui insert l’utilisateur dans la base de données | | |
| Résultat attendu | L’utilisateur doit avoir été insérer dans la base de données | | |

| ID / Désignation | T-001 |
| --- | --- |
| Description | L’utilisateur doit pouvoir créer un compte depuis la vue correspondante. | |
| Condition préalable au test | Les fonctionnalités d’inscription doivent avoir été remplie auparavant. | |
| Étapes du test | Etape 1 : créer un utilisateur pour pouvoir l’envoyer au contrôleur  Etape 2 : envoyer l’utilisateur au contrôleur  Etape 3 : La méthode qui insert l’utilisateur dans la base de données | |
| Résultat attendu | L’utilisateur doit avoir été insérer dans la base de données | |

| ID / Désignation | T-001 |
| --- | --- |
| Description | L’utilisateur doit pouvoir créer un compte depuis la vue correspondante. | |
| Condition préalable au test | Les fonctionnalités d’inscription doivent avoir été remplie auparavant. | |
| Étapes du test | Etape 1 : créer un utilisateur pour pouvoir l’envoyer au contrôleur  Etape 2 : envoyer l’utilisateur au contrôleur  Etape 3 : La méthode qui insert l’utilisateur dans la base de données | |
| Résultat attendu | L’utilisateur doit avoir été insérer dans la base de données | |

| ID / Désignation | T-001 |
| --- | --- |
| Description | L’utilisateur doit pouvoir créer un compte depuis la vue correspondante. | |
| Condition préalable au test | Les fonctionnalités d’inscription doivent avoir été remplie auparavant. | |
| Étapes du test | Etape 1 : créer un utilisateur pour pouvoir l’envoyer au contrôleur  Etape 2 : envoyer l’utilisateur au contrôleur  Etape 3 : La méthode qui insert l’utilisateur dans la base de données | |
| Résultat attendu | L’utilisateur doit avoir été insérer dans la base de données | |

| ID / Désignation | T-001 |
| --- | --- |
| Description | L’utilisateur doit pouvoir créer un compte depuis la vue correspondante. | |
| Condition préalable au test | Les fonctionnalités d’inscription doivent avoir été remplie auparavant. | |
| Étapes du test | Etape 1 : créer un utilisateur pour pouvoir l’envoyer au contrôleur  Etape 2 : envoyer l’utilisateur au contrôleur  Etape 3 : La méthode qui insert l’utilisateur dans la base de données | |
| Résultat attendu | L’utilisateur doit avoir été insérer dans la base de données | |

| ID / Désignation | T-001 |
| --- | --- |
| Description | L’utilisateur doit pouvoir créer un compte depuis la vue correspondante. | |
| Condition préalable au test | Les fonctionnalités d’inscription doivent avoir été remplie auparavant. | |
| Étapes du test | Etape 1 : créer un utilisateur pour pouvoir l’envoyer au contrôleur  Etape 2 : envoyer l’utilisateur au contrôleur  Etape 3 : La méthode qui insert l’utilisateur dans la base de données | |
| Résultat attendu | L’utilisateur doit avoir été insérer dans la base de données | |

| ID / Désignation | T-001 |
| --- | --- |
| Description | L’utilisateur doit pouvoir créer un compte depuis la vue correspondante. | |
| Condition préalable au test | Les fonctionnalités d’inscription doivent avoir été remplie auparavant. | |
| Étapes du test | Etape 1 : créer un utilisateur pour pouvoir l’envoyer au contrôleur  Etape 2 : envoyer l’utilisateur au contrôleur  Etape 3 : La méthode qui insert l’utilisateur dans la base de données | |
| Résultat attendu | L’utilisateur doit avoir été insérer dans la base de données | |

| ID / Désignation | T-001 |
| --- | --- |
| Description | L’utilisateur doit pouvoir créer un compte depuis la vue correspondante. | |
| Condition préalable au test | Les fonctionnalités d’inscription doivent avoir été remplie auparavant. | |
| Étapes du test | Etape 1 : créer un utilisateur pour pouvoir l’envoyer au contrôleur  Etape 2 : envoyer l’utilisateur au contrôleur  Etape 3 : La méthode qui insert l’utilisateur dans la base de données | |
| Résultat attendu | L’utilisateur doit avoir été insérer dans la base de données | |

Tableau 7: Description des cas de test

# Plan des tests

| No | Activité | Responsable | Collaboration | Délai |
| --- | --- | --- | --- | --- |
| 1 | Création de compte | Sébastien Berger | Tim Allemann | 28.03.2020 |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Tableau 8: Plan des tests

Abréviations et glossaire

| Abréviation / Terme technique | Explication |
| --- | --- |
| HERMES | La méthode de gestion de projets et de programmes HERMES 5 est une norme eCH. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Tableau 9: Abréviations et glossaire

Table des matières

Suivi des modifications 1

Description 1

1 Objectifs des tests 2

2 Stratégie en matière de tests et niveaux de tests 2

3 Objets à tester 2

4 Types de test 2

5 Couverture du test 3

5.1 Aperçu des cas de test 3

5.2 Évaluation des objectifs et couverture du test 3

6 Cadre de test 3

6.1 Conditions du test 3

6.2 Classification des défauts 3

6.3 Conditions de début et d’arrêt 4

7 Environnement de test 4

8 Infrastructure de test 4

8.1 Système de test 4

8.2 Données de test 4

8.3 Utilitaires de test 4

9 Organisation des tests 4

10 Description des cas de test 4

11 Plan des tests 4

Abréviations et glossaire 4

Table des matières 4

Table des tableaux 4

Table des tableaux

Tableau 1: Contrôle des modifications 1

Tableau 2: Objectifs généraux des tests 2

Tableau 3: Objets à tester 2

Tableau 4: Types de test 2

Tableau 5: Couverture du test 3

Tableau 6: Catégories de défauts 3

Tableau 7: Description des cas de test 4

Tableau 8: Plan des tests 4

Tableau 9: Abréviations et glossaire 4