Adrien MAZET

Glenn LOUARN

# Cahier de test -Tests Logiciels

## Spécifications des deux fonctionnalités à développer

### Connexion au chat

1. L’utilisateur lance Felix.
2. Felix affiche la fenêtre de connexion
3. L’utilisateur clique sur le bouton de connexion
4. Felix affiche un message de connexion
5. Felix initie la connexion à Camix
6. Felix ferme la vue de connexion
7. Felix affiche la vue du chat

**Variante [Modification IP]**

3.a.1L’utilisateur modifie la valeur de l’IP

3.a.2Retour à l’étape 3

**Variante [Modification Port]**

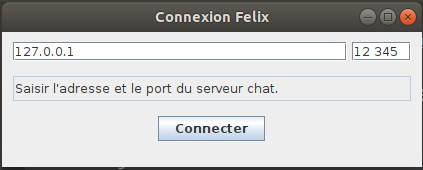
3.b.1L’utilisateur modifie la valeur du Port

3.b.2Retour à l’étape 3

**Variante [Connexion impossible]**

3.c.1Felix affiche un message connexion impossible

3.c.2Retour à l’étape 3



### Sortie du chat

1. L’utilisateur lance Felix.
2. L’utilisateur se connecte au chat
3. L’utilisateur utilise le chat
4. L’utilisateur tape la commande /q dans le chat
5. Un message de sortie du chat est affiché à l’utilisateur
6. Un message informe les autres utilisateurs de la sortie du chat de l’utilisateur
7. Le client est désinscrit du canal ou il se trouve
8. La connexion du client au chat est fermée.

## Scénarios de tests des deux fonctionnalités à développer

### Connexion au chat

|  |  |  |
| --- | --- | --- |
| **Scénario 1 : nominal** | | |
| Action | Résultats internes | Résultats externes |
| 1. L’utilisateur lance Félix  3. L’utilisateur clique sur le bouton de connexion | 5. Felix initie la connexion à Camix | 2. Felix affiche la fenêtre de connexion  4. Felix affiche un message de connexion  6. Felix ferme la vue de connexion  7. Felix affiche la vue du chat |

|  |  |  |
| --- | --- | --- |
| **Scénario 2 : modification IP et Port et connexion réussie** | | |
| Action | Résultats internes | Résultats externes |
| 1. L’utilisateur lance Félix  3.a.1 L’utilisateur modifie la valeur de l’IP  3.b.1 L’utilisateur modifie la valeur du Port  3. L’utilisateur clique sur le bouton de connexion | 5. Felix initie la connexion à Camix | 2. Felix affiche la fenêtre de connexion  4. Felix affiche un message de connexion  6. Felix ferme la vue de connexion  7. Felix affiche la vue du chat |

|  |  |  |
| --- | --- | --- |
| **Scénario 3 : connexion impossible** | | |
| Action | Résultats internes | Résultats externes |
| 1. L’utilisateur lance Félix  3. L’utilisateur clique sur le bouton de connexion  3.c.2 Felix affiche un message connexion impossible |  | 2. Felix affiche la fenêtre de connexion |

|  |  |  |
| --- | --- | --- |
| **Scénario 4 : modification IP et connexion impossible** | | |
| Action | Résultats internes | Résultats externes |
| 1. L’utilisateur lance Félix  3.a.1 L’utilisateur modifie la valeur de l’IP  3. L’utilisateur clique sur le bouton de connexion  3.c.2 Felix affiche un message connexion impossible |  | 2. Felix affiche la fenêtre de connexion |

|  |  |  |
| --- | --- | --- |
| **Scénario 5 : modification Port et connexion impossible** | | |
| Action | Résultats internes | Résultats externes |
| 1. L’utilisateur lance Félix  3.b.1 L’utilisateur modifie la valeur du Port  3. L’utilisateur clique sur le bouton de connexion  3.c.2 Felix affiche un message connexion impossible |  | 2. Felix affiche la fenêtre de connexion |

### Sortie du chat

|  |  |  |
| --- | --- | --- |
| **Scénario 1 : nominal** | | |
| Action | Résultats internes | Résultats externes |
| 1. L’utilisateur lance Felix.  2. L’utilisateur se connecte au chat  3. L’utilisateur utilise le chat  4. L’utilisateur tape la commande /q dans le chat | 7. Le client est désinscrit du canal ou il se trouve  8. La connexion du client au chat est fermée. | 5. Un message de sortie du chat est affiché à l’utilisateur  6. Un message informe les autres utilisateurs de la sortie du chat de l’utilisateur |

## Résultats de l’exécution des tests

### Tests de validation

|  |  |  |
| --- | --- | --- |
| **Intitulé** | **Prérequis** | **Résultat** |
| Connexion impossible | Camix à l’arrêt | Test passed |
| Connexion impossible Port modifié | Camix à l’arrêt | Test passed |
| Connexion impossible IP modifiée | Camix à l’arrêt | Test passed |
| Connexion impossible IP et Port modifiés |  | Test failed |
| Connexion possible | Camix lancé | Test passed |
| Connexion possible avec plusieurs clients | Camix lancé | Test passed |

### Tests d’intégration

|  |  |  |
| --- | --- | --- |
| **Intitulé** | **Prérequis** | **Résultat** |
| Ajout client non présent EasyMock | Aucun | Test passed |
| Ajout client non présent Mockito | Aucun | Test passed |
| Informe départ client | Aucun | Test failed |

### Tests unitaires

|  |  |  |
| --- | --- | --- |
| **Intitulé** | **Prérequis** | **Résultat** |
|  |  |  |
|  |  |  |
|  |  |  |

### Résultats globaux

|  |  |  |  |
| --- | --- | --- | --- |
| **Tests de validation** | **Tests d’intégration** | **Tests unitaire** | **Total** |
| 0/X | 0/Y | 0/Z | 0/X+Y+Z |

## Bonus: Retour d’experience sur le test API Rest

Test d’une API Rest déployé sur le cloud avec Firebase est une base de données Firestore.

Difficultés : les tests utilisent l’API REST et modifient donc les données en base, il aurait fallu créer une base de données spécialement pour les tests.

Pour tester un POST j’utilise un GET et vice versa. Je fais donc le postulat que au moins un des deux marches.

Test asynchrones écrivant sur la même base de données : les tests doivent être lancés à la suite et non en parallèle pour éviter des écritures simultanées et des résultats incohérents.

Language : Typescript

Framework de test : Jest (utilisé pour faire des assertions sur les résultats des requêtes à mon API).