## Exemple de concept de test appliqué à un projet réel.

Une **application bancaire en ligne**. Cet exemple détaille les étapes nécessaires à la mise en place d'un concept de test, couvrant les **objectifs** de test, **l'organisation**, **l'infrastructure**, et plus encore.

**Concept de Test pour une Application Bancaire en Ligne**

# 1 Introduction

L'application bancaire en ligne permet aux utilisateurs de gérer leurs comptes courants, d'effectuer des virements, de consulter leurs relevés de compte, et de gérer leurs investissements. L'objectif du concept de test est de s'assurer que toutes les fonctionnalités fonctionnent comme prévu et que l'application est sécurisée, performante, et conforme aux exigences.

# 2 Objectifs de test

- **Test fonctionnel** : Vérifier que les fonctionnalités de base (virements, gestion des comptes) fonctionnent correctement.
- **Test de sécurité** : S'assurer que les transactions et les données sensibles sont protégées contre toute tentative d'intrusion ou d'accès non autorisé.
- **Test de performance** : Vérifier que l'application répond correctement sous charge (ex. : utilisation simultanée par plusieurs utilisateurs).
- **Test de compatibilité** : Tester l'application sur différents navigateurs et appareils mobiles (iOS, Android).
- **Test d'utilisabilité** : Vérifier que l'interface utilisateur est intuitive et permet aux utilisateurs de réaliser des opérations sans difficultés.

# 3 Objets à tester

- **Fonctionnalités principales** :
    - Connexion sécurisée (authentification à deux facteurs).
    - Consultation du solde des comptes courants et épargne.
    - Virements internes et externes.
    - Consultation et téléchargement des relevés bancaires.
    - Gestion des placements (achat/vente d'actions).
- **Modules supplémentaires** :
    - Notifications (SMS, emails).
    - Personnalisation des alertes (ex. : notification de solde bas).
- **API** :
    - API de communication avec les systèmes de paiement et les partenaires externes (virements, gestion des investissements).

# 4 Types de tests

- **Tests fonctionnels** :
    - Vérification que les utilisateurs peuvent se connecter, afficher leurs soldes, effectuer des virements, et consulter leurs relevés sans erreurs.
- **Tests de sécurité** :
    - Utilisation d'outils de test de pénétration pour s'assurer qu'il n'y a pas de failles dans le système d'authentification.
    - Vérification que les données sensibles sont chiffrées.
- **Tests de performance** :
    - Utilisation de **JMeter** pour simuler un grand nombre d'utilisateurs et vérifier que le système reste réactif sous charge.
- **Tests de compatibilité** :
    - Exécution de tests sur plusieurs navigateurs (Chrome, Firefox, Safari) et sur les appareils mobiles (iOS et Android).
- **Tests de régression** :
    - Vérification que les nouvelles versions de l'application ne brisent pas les fonctionnalités existantes après chaque mise à jour.

# 5 Infrastructure de test

- **Environnement de développement** :
    - Serveurs locaux pour exécuter les premières phases de test (unitaires et d'intégration).
- **Environnement de test** :
    - Serveur de test dédié hébergeant une copie des bases de données de production (données anonymisées) et les versions de test des API.
    - Utilisation de machines virtuelles pour simuler différents systèmes d'exploitation et navigateurs (Windows, macOS, iOS, Android).
- **Outils** :
    - **Selenium** pour les tests automatisés d'interface utilisateur.
    - **JMeter** pour les tests de charge et de performance.
    - **OWASP ZAP** pour les tests de sécurité.
    - **Postman** pour le test des API.
    - **Jira** pour la gestion des bugs et le suivi des anomalies.

# 6 Organisation des tests

- **Équipe de test** :
    - **Testeurs fonctionnels** : Responsable de la vérification des fonctionnalités (connexion, virements, relevés).
    - **Testeurs de sécurité** : Responsable de la validation des aspects de sécurité.
    - **Testeurs de performance** : Responsable des tests de charge et de performance.
    - **Testeurs de compatibilité** : Testent l'application sur différents appareils et navigateurs.
- **Responsabilités** :
    - Chaque testeur est responsable de l'exécution des tests et de la création de rapports sur les bugs détectés.
    - Les bugs critiques doivent être priorisés et affectés aux développeurs pour correction.
- **Processus de gestion des anomalies** :
    - Les anomalies sont enregistrées dans Jira, classées par priorité (critique, élevée, moyenne, faible) et assignées à un développeur pour correction.
    - Les anomalies sont testées de nouveau après correction et validées avant de fermer le ticket.

# 7 Plan de test

- **Phase 1 : Tests fonctionnels**
    - Tester les fonctionnalités de base de l'application (connexion, virements, consultation des relevés) sur différents appareils et navigateurs.
    - Durée : 2 semaines.
- **Phase 2 : Tests de sécurité**
    - Utilisation d'outils de test de pénétration pour simuler des attaques et vérifier la résistance du système.
    - Durée : 1 semaine.
- **Phase 3 : Tests de performance**
    - Simuler plusieurs milliers d'utilisateurs simultanés pour tester la réactivité de l'application.
    - Durée : 1 semaine.
- **Phase 4 : Tests de compatibilité**
    - Tester l'application sur différentes configurations (appareils, systèmes d'exploitation, navigateurs).
    - Durée : 1 semaine.
- **Phase 5 : Tests de régression**

- Vérifier que les corrections de bugs et les nouvelles fonctionnalités n'ont pas introduit de régressions.
- Durée : 1 semaine.

## 8 Critères d'entrée et de sortie

- **Critères d'entrée** :
  - Le développement des fonctionnalités est terminé et les environnements de test sont prêts.
  - Les testeurs ont accès aux données de test et aux outils nécessaires.
- **Critères de sortie** :
  - Tous les cas de test critiques ont été validés sans erreur.
  - Les tests de sécurité n'ont pas détecté de failles critiques.
  - Les tests de performance ont démontré que l'application reste réactive même sous charge importante.
  - Aucun bug bloquant ou critique n'est ouvert.

## 9 Livrables

- **Rapports de test** : Document détaillant les résultats de chaque phase de test (fonctionnels, performance, sécurité).
- **Rapport de régression** : Rapport final après la phase de régression, incluant un état des anomalies résolues et ouvertes.
- **Plan de déploiement** : Un document final indiquant que l'application est prête pour la mise en production, après validation des tests.

## 10 Risques identifiés

- **Retard dans la livraison** : Si des bugs critiques sont découverts tardivement, cela pourrait retarder la mise en production.
- **Problèmes de compatibilité** : Certaines anciennes versions de navigateurs pourraient ne pas être pleinement compatibles.
- **Risques de performance** : En cas de surcharge importante, l'application pourrait ralentir, ce qui pourrait nécessiter des optimisations supplémentaires.

**Résumé :**

Cet exemple de **concept de test** pour une application bancaire en ligne montre comment structurer un processus de test pour un projet réel. Il couvre toutes les étapes, des tests fonctionnels aux tests de sécurité, tout en s'assurant que l'infrastructure, l'organisation et les risques sont correctement pris en compte.

Ce type de concept est applicable à divers types de projets avec quelques ajustements en fonction des exigences spécifiques et de la nature de l'application.