

Modèle de stratégie de test

Kit Ismail 3000 · version 1.0 · à copier et compléter. Les passages en italique sont des consignes à remplacer.

1. Contexte

Élément	À compléter
Produit	<i>Nom du produit et description en une phrase</i>
Utilisateurs	<i>Qui l'utilise, combien, sur quels appareils</i>
Équipe	<i>Nombre de devs, QA, PO ; rythme de livraison</i>
Enjeux	<i>Ce qui coûte cher en cas de bug : argent, données, image, réglementaire</i>
Date et version du document	

2. Objectifs qualité

Trois à cinq objectifs mesurables, alignés sur les enjeux.

- Aucun bug bloquant sur les parcours critiques en production.
- Une régression détectée avant la mise en production dans *X* % des cas.
- Un retour des tests en moins de *X* minutes à chaque pull request.

3. Périmètre

Dans le périmètre	Hors périmètre
<i>Applications, modules, API, plateformes testés</i>	<i>Ce qui ne sera pas testé, et pourquoi</i>

4. Analyse des risques

Listez les fonctionnalités, notez la probabilité et l'impact de 1 à 3, puis multipliez. Les risques notés 6 et 9 sont prioritaires.

Fonctionnalité	Probabilité (1-3)	Impact (1-3)	Score	Priorité
<i>Paiement</i>				
<i>Inscription / connexion</i>				

5. Parcours critiques

Les parcours qui ne doivent jamais casser. Ils sont testés à chaque livraison et automatisés en premier.

1. *Parcours 1*
2. *Parcours 2*
3. *Parcours 3*

6. Niveaux et types de test

Niveau	Qui	Outils	Quand
Tests unitaires	Devs	<i>Jest, JUnit...</i>	À chaque commit
Tests d'API / intégration	Devs, QA	<i>Postman, Karate, Playwright API...</i>	À chaque pull request
Tests de bout en bout	QA	<i>Playwright, Cypress, Maestro...</i>	À chaque pull request (parcours critiques), chaque nuit (suite complète)
Tests exploratoires	QA, PO	Sessions chronométrées	Chaque sprint
Non fonctionnel	<i>Qui</i>	<i>k6, Lighthouse, axe...</i>	<i>Fréquence</i>
Tests des fonctionnalités IA	QA	<i>Jeu de scénarios, LLM évaluateur</i>	À chaque changement de prompt ou de modèle

7. Automatisation

- **Ce qu'on automatise d'abord** : les parcours critiques et les régressions déjà rencontrées.
- **Ce qu'on n'automatise pas** : *écrans très instables, tests ponctuels...*
- **Intégration CI** : *outil de CI, déclencheurs, temps d'exécution cible.*
- **Règle de stabilité** : un test instable est réparé ou retiré de la suite bloquante sous *X* jours.

8. Environnements et données de test

Environnement	Usage	Données
<i>Recette / staging</i>	<i>Tests de bout en bout et recette</i>	<i>Jeux de données anonymisés, comptes de test</i>

9. Critères d'entrée et de sortie

- **Une fonctionnalité entre en test quand** : *critères d'acceptation écrits, build déployé, tests unitaires verts.*

- **Une version part en production quand** : *zéro bug bloquant ouvert, parcours critiques verts, checklist de recette déroulée.*

10. Gestion des anomalies

- Outil : *Jira, Linear, GitHub Issues...*
- Modèle de ticket : étapes, résultat obtenu, résultat attendu, environnement, capture.
- Sévérités : **Bloquant** (corrigé avant livraison), **Majeur** (corrigé dans le sprint), **Mineur** (priorisé au backlog).

11. Indicateurs suivis

Indicateur	Cible	Fréquence
Couverture des parcours critiques automatisés	<i>X %</i>	Chaque sprint
Taux de tests instables	<i>< X %</i>	Chaque sprint
Durée de la suite de tests en CI	<i>< X min</i>	Chaque sprint
Bugs trouvés en production	<i>Tendance à la baisse</i>	Chaque mois

12. Rôles

Rôle	Responsabilités
<i>QA</i>	<i>Stratégie, automatisation, recette, indicateurs</i>
<i>Devs</i>	<i>Tests unitaires, correction, revue des tests</i>
<i>PO</i>	<i>Critères d'acceptation, validation fonctionnelle</i>

Ce modèle est libre d'utilisation dans vos projets. Retrouvez d'autres kits et des articles sur la QA augmentée par l'IA sur ismail3000.