Plan de Test

EasyBooking

Plateforme de réservation de salles

TP Tests Logiciels 2024-2025

1. Introduction

1.1 Objectif du document

Ce document décrit la stratégie de test pour l'application EasyBooking, une plateforme de réservation de salles développée dans le cadre du TP « Tests Logiciels ».

1.2 Portée

Le plan couvre les tests de l'application web EasyBooking incluant :

• Interface utilisateur (Next.js)

• API Backend (Next.js API Routes)

• Base de données (Supabase/PostgreSQL)

• Authentification (Supabase Auth)

1.3 Références

• PRD.md - Document des exigences produit

• Consignes TP Tests Logiciels 2024-2025

2. Contexte du Projet

2.1 Description de l'application

EasyBooking est une application web permettant aux utilisateurs de :

• Créer un compte et se connecter

• Consulter la liste des salles disponibles

• Réserver une salle pour un créneau horaire

• Gérer ses réservations (consulter, annuler)

2.2 Stack technique

Composant	Technologie
Frontend	Next.js 14, React 18, TypeScript
Styling	Tailwind CSS 3
Backend	Next.js API Routes
Base de données	Supabase (PostgreSQL)
Authentification	Supabase Auth
Validation	Zod

3. Stratégie de Test

3.1 Niveaux de test

Niveau	Description	Outils	Responsable
Tests Unitaires	Validation des composants isolés	Jest, RTL	Développeur
Tests d'Intégration	Validation interactions modules	Jest, node-mocks-http	Développeur
Tests E2E	Validation parcours complets	Playwright	QA
Tests Performance	Validation charge et temps	k6	QA

3.2 Critères d'entrée

- Code source disponible et compilable
- Environnement de test configuré
- Base de données initialisée avec données de test
- Documentation des fonctionnalités disponible

3.3 Critères de sortie

- 100% des tests planifiés exécutés
- Taux de réussite > 90%
- Aucun bug bloquant ou critique non résolu
- Rapport de test généré

4. Périmètre des Tests

4.1 Fonctionnalités testées

ID	Fonctionnalité	Priorité	Types de tests
F1	Inscription utilisateur	Haute	Unit, Intégration, E2E
F2	Connexion utilisateur	Haute	Unit, Intégration, E2E
F3	Liste des salles	Haute	Unit, Intégration, E2E, Perf
F4	Réservation de salle	Haute	Unit, Intégration, E2E
F5	Gestion réservations	Moyenne	Unit, Intégration, E2E
F6	Navigation et UI	Moyenne	E2E
F7	Responsive design	Basse	E2E

4.2 Fonctionnalités non testées

- Administration système
- Envoi d'emails (notifications)
- Intégration calendrier externe

5. Types de Tests

5.1 Tests Unitaires (42 cas)

Objectif : Valider le comportement isolé des composants et fonctions.

Fichier : tests/unit/unit.test.tsx

Catégorie	Nombre	Description
Validation Auth	12	Schémas login et register (Zod)
Validation Booking	8	Schéma de réservation
Utilitaires	14	Fonctions cn, formatTime, formatDate
Composants UI	8	Button, Input, Badge, Alert

Commande : npm run test:unit

5.2 Tests d'Intégration (58 cas)

Objectif : Valider les interactions entre les modules et les API.

Module	Tests	Description
Rooms API	16	CRUD salles, filtrage, validation
Bookings API	20	CRUD réservations, conflits, validation
Auth API	22	Inscription, connexion, validation

Commande : npm run test:integration

5.3 Tests E2E (32 cas)

Objectif : Valider les parcours utilisateur complets dans un navigateur.

Catégorie	Tests	Description
Page d'accueil	3	Affichage, navigation
Formulaire inscription	5	Validation, erreurs, soumission
Formulaire connexion	5	Validation, erreurs, soumission
Accessibilité	2	Labels, navigation clavier
Navigation	5	Routes, redirections, 404
UI Elements	4	Header, footer, sections
Responsive	3	Mobile, tablette
Performance	3	Temps de chargement
Sécurité	2	Masquage mot de passe

Commande : npm run test:e2e

5.4 Tests de Performance (15 cas)

Objectif : Valider les performances sous charge.

Fichier : tests/performance/load-test.js

Test	Description	Seuil
Page d'accueil	Temps de chargement	< 2s
Page connexion	Temps de chargement	< 2s
Page inscription	Temps de chargement	< 2s
API Rooms	Temps de réponse	< 1.5s
API Bookings	Temps de réponse	< 1.5s
Charge multiple	Requêtes simultanées	< 2s
Headers	Latence serveur	< 1s
Erreurs 404	Gestion erreurs	< 2s
Stress login	Charge API auth	< 2s
Stress register	Charge API register	< 3s
Assets	Logo et fichiers statiques	< 500ms
Concurrence	Multi-pages aléatoires	< 2s
Bande passante	Taille réponses	< 5MB
Stabilité	Requêtes répétées	100% succès
Résilience	Gestion timeouts	< 10s

Scénario de charge :

• Montée progressive : 0 → 10 → 20 utilisateurs

• Durée totale : 3.5 minutes

• Critères : 95% des requêtes < 2s, taux d'erreur < 10%

Commande : npm run test:perf

6. Environnement de Test

6.1 Environnement matériel

Élément	Spécification
OS	macOS / Linux / Windows
RAM	8 GB minimum
CPU	Multi-core recommandé
Réseau	Connexion internet requise

6.2 Environnement logiciel

Logiciel	Version
Node.js	18+
npm	9+
Navigateur	Chromium (Playwright)
k6	1.5+

6.3 Données de test

• Base de données Supabase avec schéma initialisé

• 5 salles de test pré-configurées

• Utilisateurs de test créés dynamiquement

7. Planning des Tests

7.1 Phases d'exécution

Phase	Activité	Durée
1	Configuration environnement	30 min
2	Exécution tests unitaires	10 min
3	Exécution tests intégration	15 min
4	Exécution tests E2E	20 min
5	Exécution tests performance	15 min
6	Analyse et rapport	30 min

7.2 Commandes d'exécution

• Tous les tests (sauf performance) : **npm run test:all**

• Tests unitaires : **npm run test:unit**

• Tests intégration : **npm run test:integration**

• Tests E2E : **npm run test:e2e**

• Tests performance : **npm run test:perf:quick**

• Avec couverture : **npm run test:coverage**

8. Gestion des Anomalies

8.1 Classification des bugs

Sévérité	Description	Exemple
Bloquant	Application inutilisable	Crash au démarrage
Critique	Fonctionnalité principale KO	Impossible de réserver
Majeur	Fonctionnalité dégradée	Erreur affichage données
Mineur	Problème cosmétique	Alignement incorrect

8.2 Workflow de gestion

1. Détection du bug
2. Création d'une issue GitHub
3. Classification (sévérité, priorité)
4. Assignation au développeur
5. Correction
6. Vérification
7. Clôture

9. Livrables

Livrable	Description
Plan de test	Ce document
Fiches de test	Cas de test détaillés avec résultats
Rapport qualité	Synthèse des résultats et métriques
Code des tests	Fichiers de tests automatisés
Captures d'écran	Preuves d'exécution

10. Récapitulatif

10.1 Synthèse des tests

Type	Nombre	Outil	Fichier(s)
Unitaires	42	Jest	tests/unit/unit.test.tsx
Intégration	58	Jest	tests/integration/*.ts
E2E	32	Playwright	tests/e2e/*.spec.ts
Performance	15	k6	tests/performance/load-test.js
TOTAL	**147**		

10.2 Couverture fonctionnelle

Fonctionnalité	Unit	Intég	E2E	Perf
Inscription	✓	✓	✓	✓
Connexion	✓	✓	✓	✓
Liste salles	✓	✓	✓	✓
Réservation	✓	✓	✓	✓
Mes réservations	✓	✓	✓	✓
Navigation	-	-	✓	✓
Responsive	-	-	✓	-