



L'Application de votre Santé Mentale

CAHIER DE TESTS

BLOC 2 — Développer et tester les applications informatiques

Projet	CESIZen — Application de santé mentale
Auteur	Sam DEPARDIEU
Formation	Titre CDA — Concepteur Développeur d'Applications
Date	20 mai 2026
Version	1.0.0
Modules couverts	Comptes utilisateurs, Informations, Diagnostics
Outil d'automatisation	PHPUnit / Pest (Laravel)

1. Introduction et Contexte

Le présent document constitue le cahier de tests de l'application CESIZen, développée dans le cadre de l'Activité 2 du titre Concepteur Développeur d'Applications (CDA). Il formalise l'ensemble des scénarios de tests nécessaires à la validation de la qualité et de la conformité du prototype fonctionnel.

Ce cahier couvre trois types de tests conformément aux exigences du cahier des charges :

Tests unitaires (TU) : validation de la logique métier isolée, au niveau fonction ou classe.

Tests fonctionnels (TF) : validation des parcours utilisateurs complets via l'interface.

Tests de non-régression (TNR) : vérification que les évolutions n'impactent pas les fonctionnalités existantes.

Les modules couverts sont les deux modules obligatoires — Comptes utilisateurs et Informations — ainsi que le module Diagnostics, retenu comme module facultatif.

2. Outils d'Automatisation

L'outil d'automatisation retenu est PHPUnit, intégré nativement à Laravel via le package Pest. Ce choix est justifié par les critères suivants :

Critère	Justification
Intégration Laravel	PHPUnit est le framework de test officiel de Laravel. Pest en est une surcouche syntaxique élégante.
Tests unitaires	Permet de tester les fonctions métier (calcul du score Holmes-Rahe, logique de classification).
Tests fonctionnels	La classe TestCase de Laravel simule des requêtes HTTP complètes (GET, POST) avec session et CSRF.
Non-régression	Exécution via 'php artisan test' avant chaque merge ou déploiement. Compatible CI/CD (GitHub Actions).
Versioning intégré	PHPUnit/Pest est géré via Composer, son versionnage est donc lié au code source (composer.lock).
Communauté	Soutenu par l'équipe Laravel et une large communauté open-source. Documentation abondante.

Commande d'exécution : `php artisan test --coverage`

MODULE 1 — COMPTES UTILISATEURS — Obligatoire

Ce module gère l'ensemble du cycle de vie des comptes utilisateurs : inscription, connexion, déconnexion, gestion du profil et sécurité. Il est développé avec le contrôleur AuthController et ProfileController (Laravel).

2.1 Tests Unitaires

ID	Type	Scénario / Description	Résultat attendu	Responsable
TU-CU-01	Unitaire	Validation du format email Soumettre le formulaire d'inscription avec l'email 'invalid@' ou 'test'.	Réponse HTTP 422. Message : 'The email field must be a valid email address'.	Développeur
TU-CU-02	Unitaire	Longueur minimale du mot de passe Soumettre le formulaire avec un mot de passe de 6 caractères (ex. 'abc123').	Réponse HTTP 422. Message : 'The password field must be at least 8 characters'.	Développeur
TU-CU-03	Unitaire	Hachage bcrypt du mot de passe Vérifier le champ 'password' en base après création d'un compte.	La valeur du champ commence par '\$2y\$' (algorithme bcrypt). Le mot de passe en clair n'est jamais stocké.	Développeur
TU-CU-04	Unitaire	Unicité de l'email Tenter de créer un compte avec un email déjà présent en base.	Réponse HTTP 422. Message : 'The email has already been taken'.	Développeur
TU-CU-05	Unitaire	Confirmation de mot de passe Soumettre le formulaire avec password='MonMDP123' et password_confirmation='Autre456'.	Réponse HTTP 422. Erreur de validation sur le champ 'password_confirmation'.	Développeur

2.2 Tests Fonctionnels

ID	Type	Scénario / Description	Résultat attendu	Responsable
TF-CU-01	Fonctionnel	Inscription d'un nouvel utilisateur Précond. : email non existant. Étapes : 1) Accéder à /register 2) Remplir nom, email, mot de passe, confirmation 3) Valider.	Compte créé en BDD. Redirection vers '/' (accueil). Utilisateur automatiquement connecté.	Testeur
TF-CU-02	Fonctionnel	Inscription avec email déjà existant Précond. : email 'test@cesizen.fr' existe. Étapes : 1) Accéder à /register 2) Saisir cet email 3) Valider.	Message d'erreur affiché sous le champ email. Aucun compte créé.	Testeur

ID	Type	Scénario / Description	Résultat attendu	Responsable
			Formulaire rechargé.	
TF-CU-03	Fonctionnel	Connexion avec identifiants valides Précond. : compte 'user@cesizen.fr' / 'password123' existe. Étapes : 1) /login 2) Identifiants corrects 3) Valider.	Connexion réussie. Session régénérée. Redirection vers '/' (redirect()->intended('/')).	Testeur
TF-CU-04	Fonctionnel	Connexion avec mauvais mot de passe Précond. : compte 'user@cesizen.fr' existe. Étapes : 1) /login 2) Saisir bon email + mauvais mdp 3) Valider.	Message d'erreur : 'Les identifiants fournis ne correspondent pas à nos enregistrements'. Pas de session créée.	Testeur
TF-CU-05	Fonctionnel	Déconnexion Précond. : utilisateur connecté. Étapes : 1) Cliquer sur 'Se déconnecter' (POST /logout).	Session invalidée. Token CSRF régénéré. Redirection vers /login.	Testeur
TF-CU-06	Fonctionnel	Accès à /profile sans authentification Précond. : aucune session active. Étapes : 1) Accéder directement à /profile.	Redirection automatique vers /login (middleware 'auth' sur le groupe de routes protégées).	Testeur
TF-CU-07	Fonctionnel	Modification du profil (nom / email) Précond. : utilisateur connecté. Étapes : 1) /profile 2) Modifier le nom 3) Enregistrer (PATCH /profile).	Mise à jour en BDD (user->fill + save). Redirection vers /profile avec status='profile-updated'.	Testeur
TF-CU-08	Fonctionnel	Modification du mot de passe Précond. : utilisateur connecté. Étapes : 1) /profile 2) Saisir current_password + nouveau mdp + confirmation 3) Enregistrer (PUT /profile/password).	Nouveau mdp haché en BDD (Hash::make). Redirection vers /profile avec status='password-updated'.	Testeur
TF-CU-09	Fonctionnel	Suppression du compte Précond. : utilisateur connecté. Étapes : 1) /profile 2) Section suppression : saisir le mot de passe de confirmation 3) Valider (DELETE /profile).	Utilisateur déconnecté. Session invalidée. Token régénéré. Compte supprimé de la BDD. Redirection vers '/'.	Testeur

2.3 Tests de Non-Régression

ID	Type	Scénario / Description	Résultat attendu	Responsable
TNR-CU-01	Non-régression	Connexion après mise à jour du système d'authentification Réf. TF-CU-03. Exécuter après chaque mise à jour des dépendances Sanctum/Auth.	Le comportement est identique à l'avant mise à jour. Code HTTP 200. Redirection correcte.	Développeur
TNR-CU-02	Non-régression	Mots de passe existants restent valides après migration Réf. TU-CU-03. Exécuter après toute modification du système de hachage.	Tous les utilisateurs existants peuvent se connecter avec leur mot de passe habituel.	Développeur

MODULE 2 — INFORMATIONS — Obligatoire

Ce module assure l'affichage des pages de contenus publics de l'application (page d'accueil, navigation) accessibles aux visiteurs anonymes et aux utilisateurs connectés.

3.1 Tests Unitaires

ID	Type	Scénario / Description	Résultat attendu	Responsable
TU-INF-01	Unitaire	Route '/' retourne HTTP 200 Envoyer une requête GET sur '/' sans authentification.	HTTP 200. Vue 'welcome' rendue par DashboardController::index() sans erreur (compact('user', 'stats')).	Développeur
TU-INF-02	Unitaire	Page d'accueil accessible sans session Envoyer une requête GET sur '/' avec un client non authentifié.	HTTP 200. Aucune redirection vers /login. La variable \$user est null (Auth::user() non connecté).	Développeur

3.2 Tests Fonctionnels

ID	Type	Scénario / Description	Résultat attendu	Responsable
TF-INF-01	Fonctionnel	Page d'accueil — visiteur anonyme Précond. : aucune session. Étapes : 1) Ouvrir '/'. 1) Ouvrir '/'.	Page affichée avec contenu de présentation CESIZen. Liens vers /diagnostics, /relaxation visibles.	Testeur
TF-INF-02	Fonctionnel	Page d'accueil — utilisateur connecté Précond. : utilisateur	Éléments supplémentaires visibles (lien vers /profile, /emotions). Navigation enrichie.	Testeur

ID	Type	Scénario / Description	Résultat attendu	Responsable
		connecté. Étapes : 1) Ouvrir '/'.		
TF-INF-03	Fonctionnel	Navigation entre les pages de contenu Précond. : application accessible. Étapes : 1) Cliquer successivement sur chaque lien du menu.	Chaque lien charge la page correspondante sans erreur 404. Code HTTP 200 sur toutes les routes.	Testeur
TF-INF-04	Fonctionnel	Compatibilité multi-navigateurs Précond. : Chrome, Firefox, Edge, Safari. Étapes : 1) Ouvrir '/' dans chaque navigateur.	Affichage cohérent et fonctionnel. Pas de régression CSS. Polices et images correctement chargées.	Testeur
TF-INF-05	Fonctionnel	Responsive Design Précond. : outils de développement. Étapes : 1) Tester viewports 375px, 768px, 1280px.	Layout adapté à chaque résolution. Texte lisible. Menu hamburger fonctionnel en mobile.	Testeur

3.3 Tests de Non-Régression

ID	Type	Scénario / Description	Résultat attendu	Responsable
TNR-INF-01	Non-régression	Page d'accueil accessible après refactorisation des routes Réf. TF-INF-01. Exécuter après toute modification du fichier routes/web.php.	HTTP 200. Contenu identique. Aucune erreur 500.	Développeur
TNR-INF-02	Non-régression	Aucun lien cassé après modification des routes Réf. TF-INF-03. Exécuter après renommage ou suppression de routes.	Tous les liens du menu retournent HTTP 200. Aucun 404 détecté.	Développeur

MODULE 3 — DIAGNOSTICS — Facultatif

Ce module implémente le questionnaire de stress basé sur l'Échelle de Holmes et Rahe. L'utilisateur sélectionne des événements vécus dans les 12 derniers mois, le système calcule un score total et détermine un niveau de stress (Faible / Modéré / Élevé). Les résultats sont persistés en base pour les utilisateurs connectés.

4.1 Tests Unitaires

ID	Type	Scénario / Description	Résultat attendu	Responsable
TU-DIAG-01	Unitaire	Calcul du score Holmes-Rahe Simuler la sélection d'événements avec points [100, 73, 50] et additionner.	Score total = 223. Assertion : assertEquals(223, array_sum([100, 73, 50])).	Développeur
TU-DIAG-02	Unitaire	Classification niveau 'Élevé' (score >= 300) Simuler un score de 350.	niveauStress = 'Élevé'. La condition (score >= 300) est vraie.	Développeur
TU-DIAG-03	Unitaire	Classification niveau 'Modéré' (150 <= score < 300) Simuler un score de 250.	niveauStress = 'Modéré'. La condition (score >= 150 && score < 300) est vraie.	Développeur
TU-DIAG-04	Unitaire	Classification niveau 'Faible' (score < 150) Simuler un score de 80.	niveauStress = 'Faible'. La condition (score < 150) est vraie.	Développeur
TU-DIAG-05	Unitaire	Score zéro — aucun événement sélectionné Soumettre le questionnaire sans cocher d'événement (\$request->has('events') = false).	totalPoints = 0. niveauStress = 'Faible'. Aucune exception levée.	Développeur

4.2 Tests Fonctionnels

ID	Type	Scénario / Description	Résultat attendu	Responsable
TF-DIAG-01	Fonctionnel	Affichage du questionnaire — visiteur Précond. : événements (StressEvent) en BDD. Étapes : 1) Accéder à /diagnostics (GET, route publique).	Liste des événements StressEvent::all() affichée. Cases à cocher visibles. Historique vide (\$history = []).	Testeur
TF-DIAG-02	Fonctionnel	Soumission du questionnaire avec événements Précond. : /diagnostics accessible. Étapes : 1) Cocher des événements 2) POST /diagnostics.	HTTP 302. redirect()->back()->with('score', \$totalPoints). Score et niveau affichés sur la page.	Testeur

ID	Type	Scénario / Description	Résultat attendu	Responsable
TF-DIAG-03	Fonctionnel	Enregistrement pour utilisateur connecté Précond. : utilisateur connecté. Étapes : 1) Soumettre un diagnostic (POST /diagnostics).	Entrée créée dans 'resultat_diags' : id_user, score_total, niveau_stress ('Élevé'/'Modéré'/'Faible'), event_ids, date_passage = now().	Testeur
TF-DIAG-04	Fonctionnel	Pas d'enregistrement pour visiteur anonyme Précond. : aucune session. Étapes : 1) Soumettre un diagnostic.	Score affiché. Aucune entrée créée en BDD (Auth::check() = false). Table resultat_diags inchangée.	Testeur
TF-DIAG-05	Fonctionnel	Consultation de l'historique Précond. : utilisateur connecté, diagnostics passés existants. Étapes : 1) GET /diagnostics/history (route protégée 'auth').	Liste ordonnée par date_passage DESC. Chaque entrée affiche : date, score_total, niveau_stress.	Testeur
TF-DIAG-06	Fonctionnel	Accès à /diagnostics/history sans connexion Précond. : aucune session. Étapes : 1) Accéder à /diagnostics/history.	Redirection automatique vers /login (middleware 'auth').	Testeur

4.3 Tests de Non-Régression

ID	Type	Scénario / Description	Résultat attendu	Responsable
TNR-DIAG-01	Non-régression	Score correct après ajout d'un nouvel événement Réf. TU-DIAG-01. Exécuter après insertion d'un nouvel événement en BDD.	Le calcul sur les événements existants reste correct. Aucune régression sur les tests TU-DIAG-01 à 05.	Développeur
TNR-DIAG-02	Non-régression	Affichage de l'historique après refactorisation du modèle ResultatDiag Réf. TF-DIAG-05. Exécuter après modification du modèle ou de la migration.	L'historique s'affiche correctement avec toutes les colonnes attendues. Aucun champ manquant.	Développeur

5. Synthèse et Tableau de Bord

Le tableau ci-dessous récapitule le nombre de scénarios par module et par type de test.

Module	Tests Unitaires	Tests Fonctionnels	Non-Régression	Total
Comptes utilisateurs	5	8	2	15
Informations	2	5	2	9
Diagnostics	5	6	2	13
TOTAL	12	19	6	37