DOCUMENTATION TECHNIQUE

PHASE 2 - DÉVELOPPEMENT & TESTS

Application CesiZen de Bien-être Mental

Architecture Full-Stack avec Symfony et Vue.js

Projet	**CesiZen - Application de Bien-être Mental**
Phase	Phase 2 - Développement et Tests
Développeur	Esteban RACINE
Formation	Concepteur Développeur d'Applications (CDA)
Date	01/07/2025

SOMMAIRE

1. INTRODUCTION

Ce document présente la documentation technique de la phase 2 du projet CesiZen, une application développée pour le bien-être mental dans le cadre de ma formation de Concepteur Développeur d'Applications.

L'application CesiZen a été développée dans le cadre d'une commande fictive de l'État français, visant à proposer une solution numérique innovante pour améliorer le bien-être mental des citoyens. Conçue pour être accessible à tous, CesiZen met à disposition des utilisateurs plusieurs fonctionnalités pensées pour favoriser une meilleure santé mentale au quotidien : un système de gestion de comptes sécurisé, un accès à des contenus fiables sur le bien-être, et un tracker d’émotions personnel permettant à chacun de mieux comprendre et suivre son état émotionnel.

Ce qui m'a particulièrement motivé dans ce projet, c'est de pouvoir allier mes compétences techniques à un projet qui a du sens. Cette phase 2 représente l'aboutissement de mes efforts : le développement complet de l'application et l’écriture d'un cahier de tests élaboré pour valider chaque fonctionnalité.

2. PROTOTYPE FONCTIONNEL

2.1 Architecture Générale

Pour structurer CesiZen, nous avons opté pour une architecture moderne en 3 tiers qui nous semblait la plus adaptée :

• Frontend : Développement de l'interface utilisateur avec Vue.js 3, en utilisant Vue Router pour gérer la navigation. Capacitor a aussi été integré pour que l'application puisse fonctionner sur mobile si besoin.
• Backend : Côté serveur, nous avons créé une API REST avec Symfony 7.2, sécurisée par authentification JWT et documentée avec OpenAPI.
• Base de données : Nous utilisons MySQL 8.0 avec les migrations Doctrine, ce qui me permet de gérer facilement les évolutions de structure.

Cette architecture offre une séparation claire des responsabilités, ce qui rend le code plus maintenable et permet une évolution future plus facile.

2.2 Modules Implémentés

Trois modules fonctionnels ont été developpé suite aux demandes :

Module 1 : Gestion des Comptes Utilisateurs

Une gestion des utilisateurs était nécessaires avec des droits bien définis permettant l'accès aux différentes fonctionnalités. Il y a donc 3 rôles, les utilisateurs anonymes, les utilisateurs connectés et les administrateurs qui peuvent parametrer l'application.

Leur connexion se fait par token JWT d'une validité de 1h.

Module 2 : Gestion des Informations

Des informations sur la santé en Générale peuvent être publiées sur CesiZen. N'importe quel utilisateur y a ainsi accès. Elles sont triables par menus afin de faciliter la recherche. Une partie adminiustartion a été ajoutée afin de permettre aux administrateurs de publier ces ressources.

Module 3 : Tracker d'Émotions

Un tracker d'émotion a été développé. Chaque utilisateur connecté a ainsi accès à son calendrier et peut enregistrer pour chaque jour plusieurs émotions qu'il a rencontré. Cela permet de suivre son humeur sur une longue période et de se rappeler des différentesw actions qu'il a faite grace à l'ajout de commentaire sur ses enregistrements.

Toute l’application est responsive afin de faciliter l’accès aux utilisateurs à l’application.

2.3 Technologies Utilisées

Composant	Technologie	Version
Backend Framework	Symfony	7.2
Frontend Framework	Vue.js	3.x
Base de données	MySQL	8.0
Authentification	JWT (Lexik)	2.x
ORM	Doctrine	3.x
Documentation API	Nelmio ApiDoc / OpenApi	4.x
Mobile	Capacitor	5.x

3. DOCUMENTATION TECHNIQUE

3.1 Modélisation Base de Données

Pour concevoir la base de données de CesiZen, nous avons eu une reflexion sur la structure des données pour qu'elle soit à la fois efficace et évolutive. Le Modèle Logique de Données (MLD) que créé ci dessous met l'accent sur la maintenabilité et la facilité d'évolution afin d'ajouter les différents modules qui ont été évoqués lors de nos précédentes rencontres.

Voici les entités principales dentifiées et modélisées :
• User : La gestion des utilisateurs avec un système de rôles flexible pour différents niveaux d'accès
• Info : Le stockage des articles et contenus informatifs, structuré pour faciliter la recherche et la catégorisation
• Menu : Un système de catégorisation des informations avec icone pour un meilleur design
• Tracker : L'enregistrement des émotions des utilisateurs
• Emotion : Un référentiel des émotions qui est modifiable par les administrateurs
• CategorieEmotion : Une classification des émotions pour aider les utilisateurs à mieux s'y retrouver

Le MLD (Modèle Logique des Données) complet est disponible sur la page suivante sous forme d'image.

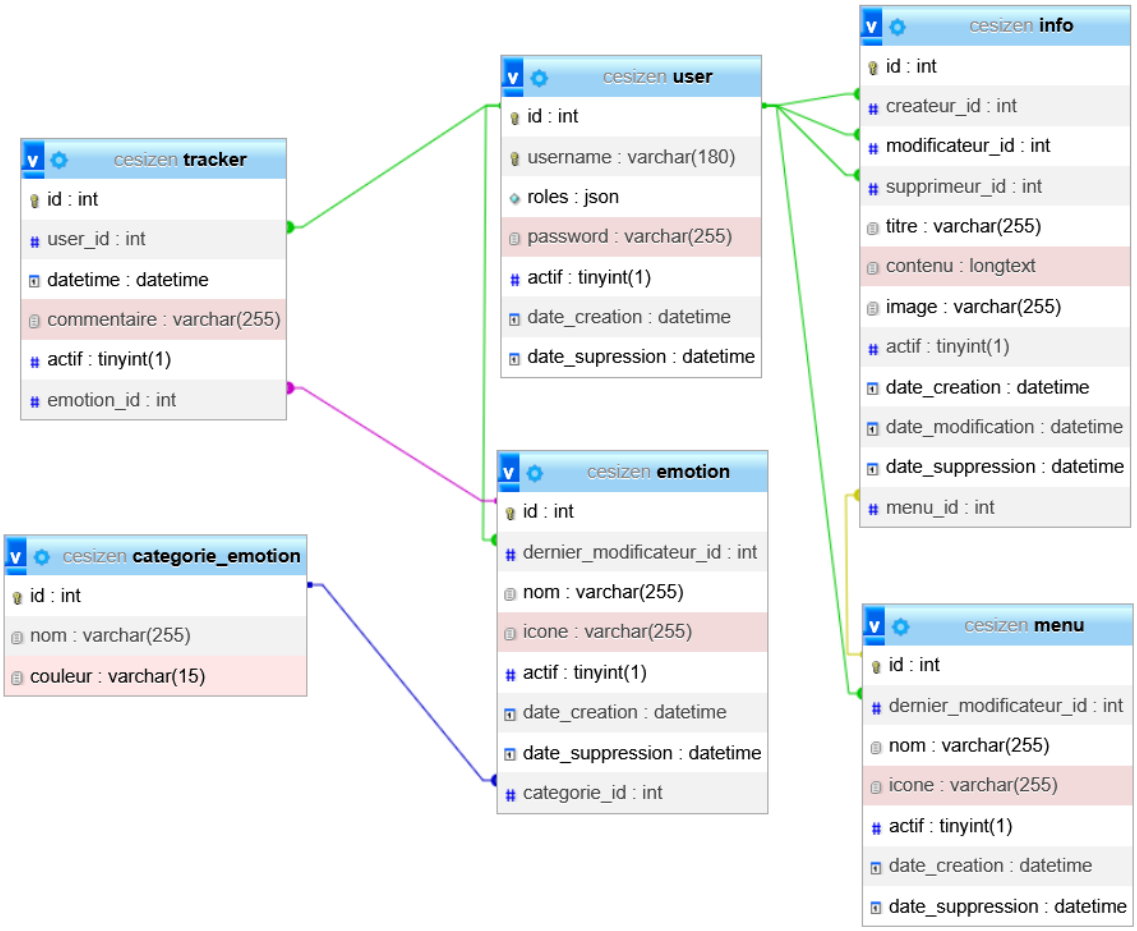

Figure : Modèle Logique de Données (MLD) de CesiZen

3.2 Comparatif des Solutions Techniques

Avant de se lancer dans le développement, nous avons pris le temps d'évaluer plusieurs solutions techniques. Ce choix était crucial car il allait déterminer la facilité de développement et la qualité finale de l'application.

SOLUTION RETENUE : Symfony + Vue.js

Pourquoi j'ai fait ce choix :

• **Expertise** : L'équipe a déjà une bonne maîtrise de Symfony acquise lors des projets précédents, ce qui nous permettait d'être efficace rapidement.

• **Robustesse** : Symfony est un framework mature avec un écosystème très riche, parfait pour une application qui doit gérer des données sensibles

• **Performances** : L'architecture API REST mise en place offre d'excellentes performances

• **Sécurité** : Avec l'authentification JWT et la validation intégrée garantit la protection des données utilisateur

• **Évolutivité** : Grâce a Doctrine, l'évolution des données de l'application est simplifiée

Les autres options qui ont été considérées :
• Symfony + Angular : Intéressant côté PHP mais courbe d'apprentissage Angular en parralèle
• Node.js + Express : Full JavaScript mais manqué d'expérience sur ces langages
• Django + Vue.js : Performant mais l'écosystème Python ne me semblait pas le plus adapté pour ce projet

Au final, le duo Symfony/Vue.js s'est imposé naturellement grâce à nos compétences existantes et à la qualité reconnue de ces frameworks.

3.3 Guide d'Installation

Le guide d'installation complet est disponible dans le fichier README.md à la racine du projet.

Prérequis :
• PHP >= 8.2 avec extensions (mysqli, pdo_mysql, openssl)
• Composer pour la gestion des dépendances PHP
• Node.js >= 18 et npm pour le frontend
• MySQL >= 8.0 pour la base de données

Installation en 4 étapes :
1. Clonage du repository et installation des dépendances
2. Configuration de l'environnement (.env.local)
3. Génération des clés JWT et création de la base
4. Démarrage des serveurs de développement

Commandes principales :
composer install && npm install
php bin/console doctrine:database:create
php bin/console doctrine:migrations:migrate
php bin/console serve -d (backend)
npm run dev (frontend)

4. CAHIER DE TESTS

4.1 Stratégie de Tests

J'ai élaboré un cahier de tests complet pour CesiZen qui comprend 52 cas de tests soigneusement répartis selon une stratégie que j'ai pensée pour être exhaustive :

Démarche de couverture fonctionnelle :
• Tests par module fonctionnel (FP1, FP2, FP3) chaque module est testé individuellement
• Tests d'intégration pour vérifier que tout fonctionne ensemble harmonieusement
• Tests d'interface pour s'assurer que l'application reste intuitive

Priorisation des tests :
• Priorité HAUTE : Les fonctionnalités critiques sans lesquelles l'app ne peut pas fonctionner
• Priorité MOYENNE : Les fonctionnalités importantes qui enrichissent l'expérience utilisateur
• Priorité FAIBLE : Les améliorations UX qui apportent une valeur ajoutée

Différents types de tests mis en place :
• Tests fonctionnels : pour valider que chaque exigence métier est respectée
• Tests d'interface : pour vérifier l'ergonomie et la fluidité de navigation
• Tests de validation : pour s'assurer que les données saisies sont correctes
• Tests transversaux : pour valider les aspects globaux de l'application

4.2 Tests par Module

Module	Nb Tests	Priorité Haute	Couverture
FP1 - Comptes Utilisateurs	12	8	Complète
FP2 - Affichage Informations	6	4	Complète
FP2 - Administration Avancée	16	6	Complète
FP3 - Tracker Émotions	11	7	Complète
Tests Transversaux	5	2	Globale
Tests Intégration	2	1	Critique

Le cahier de tests complet que j'ai créé est fourni au format Excel et contient :

• 52 tests détaillés avec des scénarios complets que j'ai rédigés pour couvrir tous les cas d'usage
• Une feuille de suivi pratique pour l'exécution des tests en conditions réelles
• Un système de codage couleur par fonctionnalité pour s'y retrouver facilement lors des tests
• Des colonnes détaillées pour chaque test : rôle requis, données d'entrée, résultats attendus, etc.

Cette approche méthodique a permis de structurer mes tests de manière claire et de ne rien oublier dans la validation de l'application.

4.3 Tests d'Intégration Automatisés

En complément des tests manuels, j'ai développé une suite de tests d'intégration automatisés avec PHPUnit qui valide le bon fonctionnement de mon API backend :

 Tests des contrôleurs implémentés :
• AuthControllerTest : Validation complète de l'authentification JWT
• UserControllerTest : Tests CRUD des utilisateurs avec gestion des rôles
• TrackerControllerTest : Tests du module de suivi émotionnel
• EmotionControllerTest : Validation de la gestion des émotions
• InfoControllerTest : Tests du module d'informations
• MenuControllerTest : Tests de la gestion des catégories
• CategorieEmotionControllerTest : Tests des catégories d'émotions

Ces tests d'intégration me permettent de :
• Valider automatiquement les endpoints de mon API
• Vérifier que l'authentification fonctionne correctement
• Tester la persistance des données en base
• M'assurer que les réponses JSON sont conformes aux attentes
• Détecter rapidement les régressions lors des modifications

L'avantage de ces tests automatisés, c'est qu'ils me font gagner un temps précieux lors des évolutions de l'application et me donnent confiance dans la stabilité du code.

5. CONCLUSION

Cette phase 2 du projet CesiZen marque l'aboutissement de plusieurs mois de travail. Nous livrons aujourd'hui un prototype fonctionnel, conforme aux objectifs donnés.

Ce que nous avons accompli :

- Développement des deux modules Comptes utilisateurs et Informations selon les spécifications.
- Intégration du module *Tracker d'Émotions*.
- Mise en place d'une architecture technique robuste et évolutive.
- Rédaction d'une documentation technique complète.
- Élaboration d'un cahier de test.

Notre valeur ajoutée :

- Interface moderne, responsive et intuitive.
- Sécurité renforcée avec JWT.
- Support mobile via Capacitor.
- Interface d'administration simple et efficace.
- Tests automatisés pour assurer la qualité du code.

Livrables du projet :

- Code source complet (Vue.js + Symfony).
- Base de données structurée avec jeu de données.
- Documentation d'installation (README).
- Cahier de tests Excel.
- Modèle de PV de recette.
- Documentation technique.

CesiZen est prêt pour la mise en production. Grâce à son architecture, l'application pourra évoluer facilement selon les besoins futurs. Nous sommes fiers du travail réalisé et convaincus de son utilité pour les utilisateurs.

Lien vers le Dépôt GitHub du projet : https://github.com/EstebanRacine/CesiZen