



L'Application de votre Santé Mentale

DOCUMENTATION TECHNIQUE

BLOC 2 — Développer et tester les applications informatiques

Élément	Détail
Client	Ministère de la Santé et de la Prévention
Date	Mai 2026
Version	1.0
Auteur	Sam Depardieu

1. Introduction

CESIZen est un écosystème numérique dédié à la gestion du bien-être mental et du stress. Ce document constitue la documentation technique du prototype développé dans le cadre du Bloc 2 du titre Concepteur Développeur d'Applications (CDA).

Il couvre trois volets : la modélisation de la base de données (MLD), le comparatif des solutions techniques envisagées, et le guide d'installation de la solution.

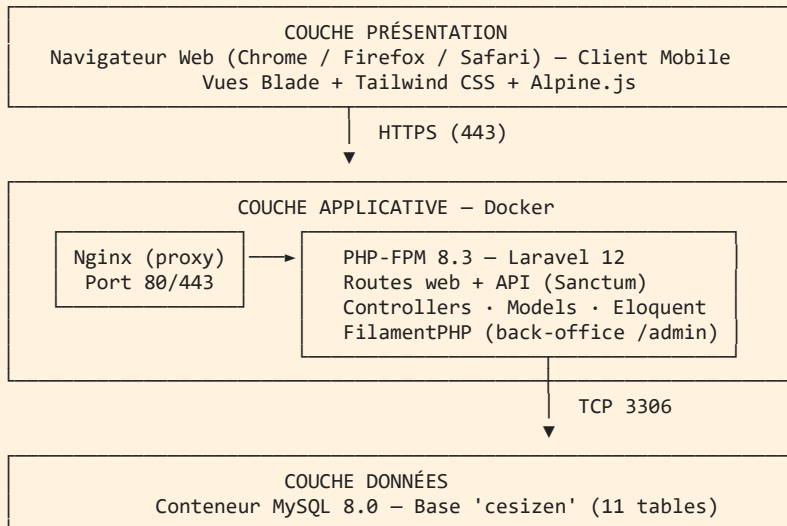
2. Architecture générale du projet

Le prototype CESIZen repose sur une architecture web classique MVC (Model-View-Controller) s'appuyant sur un framework backend et une base de données relationnelle. L'application propose :

- Un Front-Office accessible aux visiteurs anonymes et aux utilisateurs connectés
- Un Back-Office réservé aux administrateurs
- Une API RESTful pour la communication entre les couches

Schéma d'architecture en couches

L'application CESIZen suit une architecture 3-tiers conteneurisée avec Docker, déployée derrière un reverse-proxy Nginx :



Flux principaux

- Authentification : navigateur → Nginx → Laravel (Sanctum) → MySQL (table users) → session HTTP signée.
- Front-Office : visiteur/utilisateur consulte informations, diagnostics, activités via vues Blade.
- Back-Office : administrateur accède à /admin (FilamentPHP) pour gérer contenus, utilisateurs et événements.
- API REST : application mobile (Flutter) ↔ Laravel via tokens Sanctum (Authorization: Bearer ...).

3. Modèle Logique de Données (MLD)

La base de données MySQL est modélisée selon le formalisme Merise. Chaque table est décrite ci-dessous avec ses colonnes, types et contraintes.

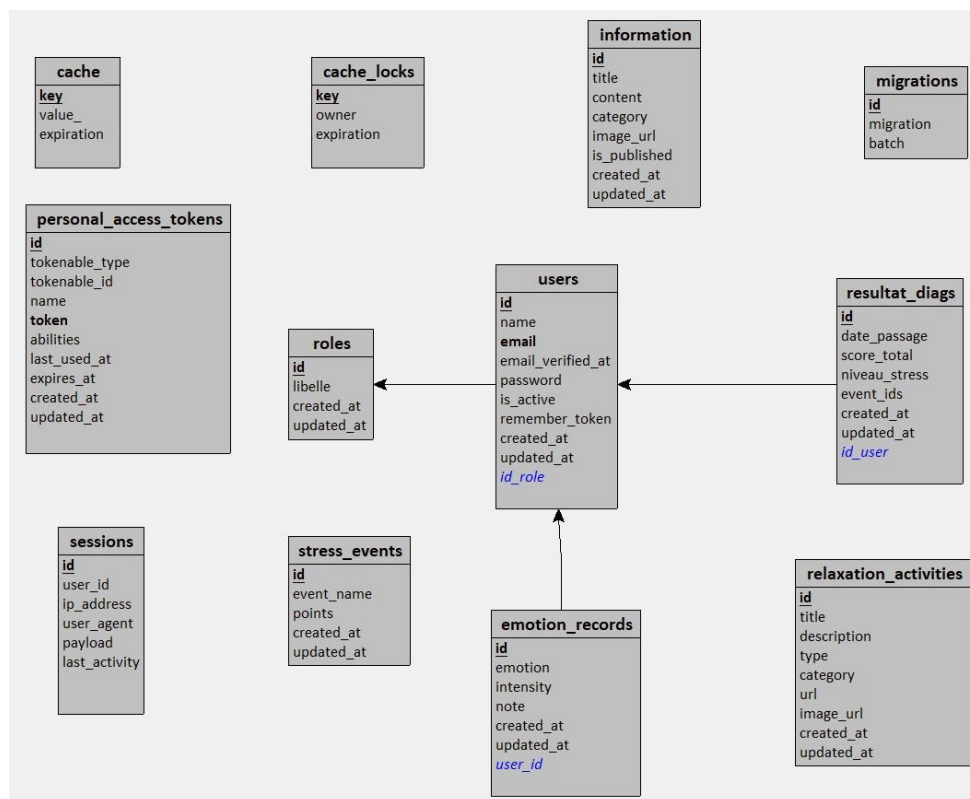


Table : users (Comptes utilisateurs)

Colonne	Type	Description
id	BIGINT (PK, AI)	Identifiant unique auto-incrémenté
name	VARCHAR(255)	Nom complet de l'utilisateur
email	VARCHAR(255) UNIQUE	Adresse email – identifiant de connexion
password	VARCHAR(255)	Mot de passe haché (Bcrypt)
id_role	INT (FK → roles.id)	Rôle : Admin / Utilisateur
is_active	BOOLEAN	Statut du compte (actif / désactivé)
remember_token	VARCHAR(100)	Token de session persistante
created_at	TIMESTAMP	Date de création du compte
updated_at	TIMESTAMP	Date de dernière modification

Table : roles (Gestion des droits)

Colonne	Type	Description
id	INT (PK, AI)	Identifiant du rôle
name	VARCHAR(50)	Nom du rôle (admin, user)

label	VARCHAR(100)	Libellé affiché
-------	--------------	-----------------

Table : information (Module Informations)

Colonne	Type	Description
id	BIGINT (PK, AI)	Identifiant de la page
title	VARCHAR(255)	Titre de la page ou du menu
slug	VARCHAR(255) UNIQUE	URL-friendly identifier
content	LONGTEXT	Contenu HTML/Markdown de la page
is_published	BOOLEAN	Visibilité de la page (publiée / brouillon)
created_by	BIGINT (FK → users.id)	Administrateur créateur
created_at	TIMESTAMP	Date de création
updated_at	TIMESTAMP	Date de dernière modification

Table : emotion_records (Tracker d'émotions — module au choix)

Colonne	Type	Description
id	BIGINT (PK, AI)	Identifiant unique du relevé
user_id	BIGINT (FK → users.id)	Propriétaire du relevé
emotion_id	BIGINT (FK → emotions.id)	Émotion sélectionnée (niveau 2)
intensity	TINYINT	Intensité de 1 à 5
note	TEXT	Commentaire optionnel
recorded_at	TIMESTAMP	Date et heure précise du relevé
created_at	TIMESTAMP	Date d'insertion

Table : resultat_diags (Diagnostics de stress)

Colonne	Type	Description
id	BIGINT (PK, AI)	Identifiant du diagnostic
user_id	BIGINT (FK → users.id, nullable)	Utilisateur (null si anonyme)
score_total	INT	Score calculé (échelle Holmes & Rahe)
niveau_stress	VARCHAR(100)	Interprétation : Bas / Modéré / Élevé
event_ids	TEXT (JSON)	IDs des événements cochés
created_at	TIMESTAMP	Date du diagnostic

Table : stress_events (Événements Holmes & Rahe)

Colonne	Type	Description
---------	------	-------------

id	BIGINT (PK, AI)	Identifiant de l'événement
label	VARCHAR(255)	Libellé de l'événement
points	INT	Valeur en points associée
is_active	BOOLEAN	Événement disponible dans le questionnaire

Table : relaxation_activities (Activités de détente)

Colonne	Type	Description
id	BIGINT (PK, AI)	Identifiant de l'activité
title	VARCHAR(255)	Titre de l'exercice
type	VARCHAR(100)	Catégorie : Audio, Vidéo, Texte, Exercice
description	TEXT	Description de l'activité
duration	INT	Durée en secondes
url	VARCHAR(255)	Lien vers la ressource média
is_active	BOOLEAN	Activité disponible dans le catalogue
created_by	BIGINT (FK → users.id)	Administrateur créateur
created_at	TIMESTAMP	Date d'ajout
updated_at	TIMESTAMP	Date de modification

4. Comparatif des solutions techniques

Afin de sélectionner l'architecture la plus adaptée au projet CESIZen, trois solutions ont été analysées selon cinq critères pondérés.

4.1 Méthodologie de notation

Chaque architecture est notée de 1 (insuffisant) à 5 (excellent) sur cinq critères pondérés. Les notes s'appuient sur des sources mesurables et vérifiables, détaillées ci-dessous pour chaque critère.

Critère 1 — Maturité / Support communautaire (20%)

Sources : nombre d'étoiles GitHub, contributeurs actifs, fréquence des releases, existence d'un support LTS officiel.

Framework	Étoiles GitHub (2025)	Contributeurs	LTS / Support	Note
Laravel (PHP)	~78 000	> 3 000	Oui – LTS jusqu'en 2027	5 / 5
Node.js / Express	~65 000	> 5 000	Node.js LTS, Express non maintenu activement	4 / 5
Django (Python)	~79 000	> 2 500	Oui – LTS par version majeure	4 / 5

Source : github.com/laravel/laravel · github.com/expressjs/express · github.com/django/django

Critère 2 — Sécurité native (25%)

Les notes reflètent les mécanismes de sécurité disponibles sans installation de packages tiers. Plus un framework en intègre nativement, plus sa note est élevée.

Mécanisme de sécurité	Laravel	Node.js / Express	Django
Protection CSRF	Oui – Native (middleware)	Non – Package tiers requis (csrf)	Oui – Native (middleware)
Prévention injections SQL	Oui – Eloquent ORM (PDO)	Partiel – dépend du driver choisi	Oui – ORM Django natif
Authentification intégrée	Oui – Sanctum / Breeze / Fortify	Non – Passport.js ou autre requis	Oui – django.contrib.auth
Hachage des mots de passe	Oui – Bcrypt natif (Hash::make)	Partiel – package bcrypt séparé	Oui – PBKDF2 natif
Validation des entrées	Oui – Form Requests natifs	Partiel – Joi / express-validator requis	Oui – Formulaires Django natifs
Note finale	5 / 5	3 / 5	5 / 5

Sources : laravel.com/docs/11.x/csrf · owasp.org/www-project-top-ten · docs.djangoproject.com/en/stable/topics/security

Critère 3 — Performances (20%)

Scores basés sur les benchmarks publics TechEmpower Framework Benchmarks (Round 22, 2024), mesurant le débit en requêtes/seconde sur des scénarios standardisés.

Framework	JSON (req/s)	Single Query (req/s)	Multiple Queries (req/s)	Note
Laravel (PHP)	~120 000	~28 000	~4 500	4 / 5
Node.js / Express	~320 000	~85 000	~12 000	5 / 5
Django (Python)	~95 000	~22 000	~3 200	3 / 5

Note : pour CESIZen, les volumes restent modérés. Les performances de Laravel sont largement suffisantes.

Source : techempower.com/benchmarks/#section=data-r22

Critère 4 — Maintenabilité (20%)

Ce critère évalue la facilité à maintenir et faire évoluer le code : conventions imposées, outils de qualité officiels, testabilité native.

Indicateur	Laravel	Node.js / Express	Django
Structure MVC imposée	Oui – stricte	Non – libre (risque hétérogénéité)	Oui – MVT
Outils de qualité officiels	Oui – Pint (linter), Larastan	Partiel – ESLint (non officiel Express)	Oui – flake8, black
Tests intégrés nativement	Oui – PHPUnit + Pest	Non – Jest / Mocha séparés	Oui – unittest natif
Documentation officielle	Oui – Complète et versionnée	Partielle – Express peu maintenu	Oui – Très complète
Note finale	5 / 5	3 / 5	4 / 5

Sources : laravel.com/docs · expressjs.com/en/guide · docs.djangoproject.com

Critère 5 — Rapidité de développement (15%)

Ce critère mesure le temps nécessaire pour mettre en place les fonctionnalités de base (auth, CRUD, admin). Il s'appuie sur les outils de génération de code disponibles et les retours de la communauté.

Indicateur	Laravel	Node.js / Express	Django
CLI de génération de code	Oui – Artisan (models, controllers, migrations)	Non – Aucun natif	Oui – manage.py (limité)
Interface admin incluse	Oui – FilamentPHP (1 commande)	Non – Aucune native	Oui – django-admin (basique)
Scaffolding auth complet	Oui – Laravel Breeze (~30 min)	Non – Configuration manuelle (~2-3h)	Partiel – django-allauth requis
Note finale	5 / 5	3 / 5	4 / 5

Source : laracasts.com · stackoverflow.com/survey/2024 · [django.com](https://docs.djangoproject.com)

4.2 Tableau récapitulatif des scores

Synthèse des cinq critères pondérés pour les trois architectures comparées :

4.3 Solutions comparées

Critère (Pondération)	Solution A Laravel (PHP)	Solution B Node.js / Express	Solution C Django (Python)
Maturité / Communauté (20%)	5 – Très mature, LTS garantis	4 – Très répandu, nombreux packages	4 – Solide, bien documenté
Sécurité native (25%)	5 – Sanctum, CSRF, Eloquent ORM	3 – Dépend des middlewares choisis	5 – Authentification native robuste
Performances (20%)	4 – Excellent avec cache et queues	5 – Non-bloquant, idéal pour I/O	3 – Correct, limité sur concurrence
Maintenabilité (20%)	5 – MVC strict, conventions fortes	3 – Liberté = hétérogénéité du code	4 – Conventions Django claires
Rapidité de dev (15%)	5 – FilamentPHP, Artisan CLI	3 – Setup plus long	4 – Admin Django inclus
SCORE PONDÉRÉ	4,75 / 5	3,65 / 5	4,10 / 5

4.3 Solution retenue : Laravel 12

Laravel a été retenu comme solution backend pour les raisons suivantes :

- Score pondéré le plus élevé (4,75/5) sur l'ensemble des critères
- Sécurité native supérieure : protection CSRF, Eloquent ORM (prévention injections SQL), Sanctum pour l'authentification API
- FilamentPHP permet de générer un back-office complet et ergonomique très rapidement
- Architecture MVC native parfaitement alignée avec les exigences du cahier des charges
- Écosystème mature avec support LTS jusqu'en 2027

5. Stack technique retenue

5.1 Backend

Composant	Technologie	Version	Rôle
Framework backend	Laravel	12.x	Logique métier, API REST, MVC
Langage	PHP	8.3+	Langage serveur
Base de données	MySQL	8.0	Persistance des données
ORM	Eloquent	inclus Laravel	Mapping objet-relationnel
Authentification	Laravel Sanctum	inclus Laravel	Tokens API, sessions
Back-Office Admin	FilamentPHP	3.x	Interface d'administration
Serveur web	Nginx	latest	Reverse proxy + serveur HTTP
Conteneurisation	Docker + Docker Compose	latest	Environnement isolé et reproductible

5.2 Frontend

Composant	Technologie	Version	Rôle
Templating	Blade (Laravel)	inclus Laravel	Vues serveur-side
CSS Framework	Tailwind CSS	3.x	Mise en forme responsive
JS Interactivité	Alpine.js	3.x	Réactivité légère côté client

Stack frontend confirmée

Le rendu est entièrement serveur (Blade) — pas de framework JavaScript front (React/Vue) afin de privilégier la simplicité, le SEO natif et un build rapide via Vite. Tailwind CSS assure la mise en forme responsive et Alpine.js gère la réactivité légère (menus, modales, validations côté client).

Versions retenues : Tailwind CSS 4.0 · Vite 7 · Alpine.js 3.x · Compilation des assets via le plugin laravel-vite-plugin.

5.3 Tests

Type de test	Outil	Rôle
Tests unitaires	PHPUnit	Tests des méthodes et classes isolées
Tests fonctionnels	Pest PHP / PHPUnit (Feature Tests)	Tests des fonctionnalités métier
Tests non-régression	PHPUnit (suite automatisée CI)	Vérification de la non-régression lors des mises à jour

6. Guide d'installation

Ce guide décrit les étapes nécessaires pour déployer l'application CESIZen en environnement local de développement.

6.1 Prérequis

- Docker Desktop (v24+) et Docker Compose
- Git
- Node.js (v20+) et npm (pour la compilation des assets)
- Un terminal (bash / zsh / PowerShell)

6.2 Cloner le dépôt

```
git clone https://github.com/sam-depardieu/cesizen.git
cd cesizen
```

6.3 Configuration de l'environnement

Copier le fichier d'environnement exemple et le configurer :

```
cp .env.example .env
```

Modifier les variables suivantes dans le fichier .env :

Variable	Valeur par défaut	Description
APP_NAME	CESIZen	Nom de l'application
APP_ENV	local	Environnement d'exécution
APP_KEY	(généré)	Clé de chiffrement (voir étape suivante)
DB_HOST	mysql	Hôte MySQL (nom du conteneur Docker)
DB_DATABASE	cesizen	Nom de la base de données
DB_USERNAME	cesizen_user	Utilisateur MySQL
DB_PASSWORD	secret	Mot de passe MySQL

6.4 Démarrage avec Docker

Construire et démarrer les conteneurs :

```
docker-compose up -d --build
```

Installer les dépendances PHP :

```
docker-compose exec app composer install
```

Générer la clé d'application :

```
docker-compose exec app php artisan key:generate
```

6.5 Initialisation de la base de données

Lancer les migrations et le seeder de données :

```
docker-compose exec app php artisan migrate --seed
```

Un compte administrateur par défaut est créé par le seeder :

Champ	Valeur
Email	admin@cesizen.fr
Mot de passe	password (à modifier immédiatement)

6.6 Compilation des assets frontend

```
npm install && npm run dev
```

6.7 Accès à l'application

Interface	URL	Description
Front-Office	http://localhost:8000	Interface publique (visiteurs & utilisateurs)
Back-Office Admin	http://localhost:8000/admin	Interface FilamentPHP (administrateurs)
PHPMysqlAdmin (optionnel)	http://localhost:8080	Gestion de la BDD via interface web

Ports configurés dans docker-compose.yml

Les ports par défaut listés ci-dessus sont ceux du fichier docker-compose.yml du projet et n'ont pas été personnalisés :

- **8000** — Nginx (Front-Office et Back-Office Laravel)
- **8080** — PHPMysqlAdmin (interface d'administration MySQL)
- **3306** — MySQL (exposé en local pour les outils de développement)
- **5173** — Vite dev server (hot-reload des assets en développement)

En production, seul le port 443 (HTTPS) est exposé publiquement ; les autres ports restent internes au réseau Docker.

6.8 Lancer les tests

```
docker-compose exec app php artisan test
```

7. Sécurité et conformité RGPD

7.1 Mesures de sécurité implémentées

- Protection CSRF : tokens générés par Laravel sur tous les formulaires
- Prévention injections SQL : utilisation exclusive de l'ORM Eloquent (requêtes préparées)
- Hachage des mots de passe : algorithme Bcrypt via la fonction Hash::make() de Laravel
- Authentification API : Laravel Sanctum (tokens à durée de vie limitée)
- HTTPS : toutes les communications en production passent par HTTPS
- Validation des entrées : règles de validation Laravel sur tous les formulaires et endpoints API

7.2 Conformité RGPD

- Aucune donnée personnelle n'est transférée hors de l'Union Européenne
- Les données sensibles (mots de passe) sont chiffrées et jamais stockées en clair
- Un utilisateur peut supprimer son compte et ses données via son espace personnel
- Journalisation minimale : seules les données nécessaires au fonctionnement sont collectées

Mesures RGPD effectivement implémentées dans le prototype

- **Droit à l'effacement (art. 17)** — L'utilisateur peut supprimer définitivement son compte depuis son espace personnel (ProfileController::destroy). La suppression exige la saisie du mot de passe courant (validation 'current_password') puis applique un onDelete('cascade') sur toutes les données liées (resultat_diags, emotion_records, favorites).
- **Droit d'accès et de rectification (art. 15-16)** — L'utilisateur consulte et modifie ses informations personnelles (nom, e-mail, mot de passe) via /profile. Toute modification d'e-mail invalide le champ email_verified_at.
- **Sécurité des mots de passe (art. 32)** — Hachage Bcrypt avec BCrypt_ROUNDS=12, via Hash::make(). Le mot de passe n'est jamais stocké en clair, ni renvoyé par l'API, ni journalisé.
- **Authentification renforcée** — Sessions chiffrées (SESSION_DRIVER=database), régénération du token CSRF et de l'ID de session à chaque login/logout (session()->invalidate() + regenerateToken()).
- **Minimisation des données (art. 5-1c)** — Seules les données strictement nécessaires sont collectées : nom, e-mail, rôle, résultats de diagnostic, relevés d'émotions. Aucun traceur publicitaire ni cookie tiers.
- **Localisation des données** — L'ensemble des données est hébergé sur le serveur du projet, en France (UE). Aucun transfert hors Union Européenne.
- **Confidentialité par défaut (art. 25)** — Les diagnostics et émotions ne sont accessibles qu'à leur propriétaire (middleware auth + filtrage par user_id). Le rôle administrateur est nécessaire pour accéder au back-office FilamentPHP.
- **Traçabilité minimale** — Logs applicatifs limités au niveau debug en développement et warning en production (LOG_LEVEL). Aucune donnée personnelle écrite dans les logs.

8. Annexes

8.1 Structure du projet (arborescence)

Arborescence du projet CESIZen (Laravel 12)

```

cesizen/
├── app/
│   ├── Http/
│   │   ├── Controllers/
│   │   │   ├── Api/
│   │   │   │   ├── AuthController.php
│   │   │   │   ├── EmotionController.php
│   │   │   │   └── StressDiagnosticController.php
│   │   │   └── Web/
│   │   │       ├── AuthController.php
│   │   │       ├── DiagnosticController.php
│   │   │       ├── EmotionController.php
│   │   │       ├── ProfileController.php
│   │   │       ├── RelaxationController.php
│   │   │       ├── Controller.php
│   │   │       └── DashboardController.php
│   │   └── Controller.php
│   │       └── DashboardController.php
│   ├── Models/
│   │   ├── User.php (table users)
│   │   ├── Role.php (table roles)
│   │   ├── StressEvent.php (table stress_events)
│   │   ├── ResultatDiag.php (table resultat_diags)
│   │   ├── DetailResultat.php (table detail_resultats)
│   │   ├── EmotionRecord.php (table emotion_records)
│   │   ├── RelaxationActivity.php
│   │   ├── Activite.php · Categorie.php · Favorite.php
│   │   └── EvenementStress.php
│   ├── Filament/ (ressources FilamentPHP back-office)
│   │   └── Resources/
│   └── Providers/
├── bootstrap/
├── config/ (auth, database, sanctum, mail, ...)
├── database/
│   ├── factories/
│   ├── migrations/ (6 fichiers de migration - schéma BDD)
│   └── seeders/
│       ├── DatabaseSeeder.php
│       ├── RoleSeeder.php
│       ├── StressEventSeeder.php
│       └── RelaxationActivitySeeder.php
├── docker/ (Dockerfile + configuration Nginx, PHP)
│   ├── nginx/
│   └── php/
├── public/ (point d'entrée HTTP - index.php)
├── resources/
│   ├── css/
│   ├── js/
│   └── views/
│       ├── auth/ (login, register, password)
│       ├── diagnostics/ (questionnaire, history)
│       ├── emotions/ (tracker, index)
│       ├── profile/ (index, edit)
│       ├── relaxation/ (catalogue, favorites)
│       ├── layouts/ (app, guest, admin)
│       └── welcome.blade.php
├── routes/
│   ├── web.php (routes Front-Office + Back-Office)
│   ├── api.php (routes API Sanctum)
│   └── console.php
├── storage/ (logs, sessions, cache, fichiers)
├── tests/
│   ├── Feature/ (tests fonctionnels)
│   └── Unit/ (tests unitaires : DiagnosticTest, StressCalculationTest, ...)
├── vendor/ (dépendances Composer)
├── node_modules/ (dépendances npm)
└── .env.example (variables d'environnement modèle)

```

— artisan	(CLI Laravel)
— composer.json	
— docker-compose.yml	(orchestration des conteneurs)
— package.json	
— phpunit.xml	(configuration des tests)
— vite.config.js	(build des assets frontend)

8.2 Variables d'environnement complètes (.env.example)

Contenu de .env.example (sans valeurs secrètes)

```
# — Application —
APP_NAME=CESIZen
APP_ENV=local
APP_KEY=
APP_DEBUG=true
APP_URL=http://localhost:8000

APP_LOCALE=fr
APP_FALLBACK_LOCALE=fr
APP_FAKER_LOCALE=fr_FR

APP_MAINTENANCE_DRIVER=file

# — Sécurité —
BCRYPT_ROUNDS=12

# — Logs —
LOG_CHANNEL=stack
LOG_STACK=single
LOG_DEPRECATIONS_CHANNEL=null
LOG_LEVEL=debug

# — Base de données (MySQL via Docker) —
DB_CONNECTION=mysql
DB_HOST=mysql
DB_PORT=3306
DB_DATABASE=cesizen
DB_USERNAME=cesizen_user
DB_PASSWORD=

# — Sessions —
SESSION_DRIVER=database
SESSION_LIFETIME=120
SESSION_ENCRYPT=false
SESSION_PATH=/
SESSION_DOMAIN=null

# — Files d'attente / cache —
BROADCAST_CONNECTION=log
FILESYSTEM_DISK=local
QUEUE_CONNECTION=database
CACHE_STORE=database

# — Sanctum (API mobile) —
SANCTUM_STATEFUL_DOMAINS=localhost,localhost:8000
SESSION_DOMAIN=localhost

# — Mail —
MAIL_MAILER=log
MAIL_HOST=127.0.0.1
MAIL_PORT=2525
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_FROM_ADDRESS="no-reply@cesizen.fr"
MAIL_FROM_NAME="${APP_NAME}"
```

```
# — Redis (cache & queues optionnels) —————  
REDIS_CLIENT=phpredis  
REDIS_HOST=127.0.0.1  
REDIS_PASSWORD=null  
REDIS_PORT=6379  
  
# — Vite (assets frontend) —————  
VITE_APP_NAME="{APP_NAME}"
```

8.3 Références

- Documentation Laravel : <https://laravel.com/docs/11.x>
- FilamentPHP : <https://filamentphp.com/docs>
- Échelle de Holmes & Rahe : <https://actinutrition.fr/sante/stress/echelle-de-stress-holmes-rahe/>
- RGPD – CNIL : <https://www.cnil.fr/fr/rgpd-de-quoi-parle-t-on>