

Dossier de projet

Candidat Néo Masson  
Chef de projet : Anass Benfares   
Expert 1 : Suleyman Ceran  
Expert 2 : Claude Albert Muller Theurillat

Table des matières

[1 Analyse préliminaire 4](#_Toc198736250)

[1.1 Introduction 4](#_Toc198736251)

[1.2 Objectifs 4](#_Toc198736252)

[1.2.1 Respect du modèle MVC avec Laravel 4](#_Toc198736253)

[1.2.2 Qualité et lisibilité du code 4](#_Toc198736254)

[1.2.3 Processus d’inscription et d’authentification complet 4](#_Toc198736255)

[1.2.4 Sécurité des accès et des données 4](#_Toc198736256)

[1.2.5 Validation serveur et retours utilisateurs clairs 4](#_Toc198736257)

[1.2.6 Système de gestion des thés 4](#_Toc198736258)

[1.2.7 Listes et exportation PDF 5](#_Toc198736259)

[1.3 Planification 5](#_Toc198736260)

[1.3.1 Planification initiale 6](#_Toc198736261)

[1.3.2 Méthodologie de travail 7](#_Toc198736262)

[2 Analyse / Conception 8](#_Toc198736263)

[2.1 Maquettes 8](#_Toc198736264)

[2.2 Modélisation de la base de données 13](#_Toc198736265)

[2.2.1 MCD 13](#_Toc198736266)

[2.2.2 MLD 13](#_Toc198736267)

[2.2.3 MPD 14](#_Toc198736268)

[2.3 Stratégie de test 14](#_Toc198736269)

[2.3.1 Tests fonctionnels 14](#_Toc198736270)

[2.3.2 Tests unitaires 15](#_Toc198736271)

[2.3.3 Données de test 16](#_Toc198736272)

[2.4 Risques techniques 16](#_Toc198736273)

[2.4.1 Apprentissage sur bibliothèques externes 16](#_Toc198736274)

[2.4.2 Expérience limitée en JavaScript 16](#_Toc198736275)

[2.4.3 Familiarité restreinte avec Laravel 16](#_Toc198736276)

[2.5 Dossier de conception 16](#_Toc198736277)

[2.5.1 Environnement 16](#_Toc198736278)

[2.5.2 Outils 17](#_Toc198736279)

[3 Réalisation 17](#_Toc198736280)

[3.1 Problèmes rencontrés 17](#_Toc198736281)

[3.1.1 Mode ‘’safe update’’ de MySQL 17](#_Toc198736282)

[3.1.2 Timeout de processus 18](#_Toc198736283)

[3.1.3 Menu latéral statique 18](#_Toc198736284)

[3.2 Fonctionnalités importantes 19](#_Toc198736285)

[3.2.1 Barre de Recherche, filtres et tri 19](#_Toc198736286)

[3.2.2 Connexion/authentification 25](#_Toc198736287)

[3.2.3 Listes déroulantes de la page d’ajout 25](#_Toc198736288)

[3.2.4 Exportation en PDF 29](#_Toc198736289)

[Description physique du projet 33](#_Toc198736290)

[Répertoires où le logiciel est installé : 33](#_Toc198736291)

[Liste des fichiers et description : 33](#_Toc198736292)

[3.3 Description des tests effectués 33](#_Toc198736293)

[1.2. Erreur restante 34](#_Toc198736294)

[1.3. Liste des documents fournis 34](#_Toc198736295)

[4 Conclusions 35](#_Toc198736296)

[4.1 Objectifs et Résultats 35](#_Toc198736297)

[4.2 Suites possibles pour le projet (évolutions & améliorations) 35](#_Toc198736298)

[4.2.1 Système de panier, commande et de paiement 35](#_Toc198736299)

[4.2.2 Gestion de rôles utilisateurs 36](#_Toc198736300)

[4.2.3 Multi-langue (i18n) 36](#_Toc198736301)

[4.2.4 Import de catalogue 36](#_Toc198736302)

[4.2.5 Recommandations et notation 36](#_Toc198736303)

[5 Annexes 36](#_Toc198736304)

[5.1 Aides externes 36](#_Toc198736305)

[5.2 Résumé du rapport du TPI 37](#_Toc198736306)

[5.2.1 Situation de départ 37](#_Toc198736307)

[5.2.2 Mise en œuvre 37](#_Toc198736308)

[5.2.3 Résultats 37](#_Toc198736309)

[5.3 Sources – Bibliographie 37](#_Toc198736310)

[5.4 Glossaire 38](#_Toc198736311)

[5.5 Table des illustrations 39](#_Toc198736312)

[5.6 Mise en place de l’environnement de projet 40](#_Toc198736313)

[5.6.1 Installer PHP 41](#_Toc198736314)

[5.6.2 Installer composer 41](#_Toc198736315)

[5.6.3 Installer Node.js et NPM 41](#_Toc198736316)

[5.6.4 Créer le projet Laravel 42](#_Toc198736317)

[5.6.5 Installer MySQL 42](#_Toc198736318)

[5.6.6 Installer Breeze 43](#_Toc198736319)

[5.6.7 Configurer l’environnement Laravel et se connecter à la DB 43](#_Toc198736320)

[5.6.8 Créer la base de données MySQL 43](#_Toc198736321)

[5.6.9 Installer Dompdf 44](#_Toc198736322)

[5.6.10 Migrations pour la base de données 44](#_Toc198736323)

[5.6.11 Lancement de l’application 45](#_Toc198736324)

[5.7 Journaux de travail 46](#_Toc198736325)

[5.4.1.1. Prérequis ? 46](#_Toc198736326)

# Analyse préliminaire

Dans cette partie du rapport nous verrons le cadre ainsi que le but du projet.

## Introduction

Ce projet s’inscrit dans le cadre du travail pratique individuel ([TPI](#_TPI_:_est)) réalisé en fin de formation d’informaticien à l’ETML. Il répond à la demande d’ExceptionTea SA, une entreprise lausannoise spécialisée dans les thés rares, qui souhaite disposer d’une solution personnalisée pour référencer, organiser et gérer sa collection.

L’application web sera développée avec le framework PHP Laravel, en suivant une architecture MVC rigoureuse. Elle proposera une interface moderne, responsive et intuitive, conçue avec HTML5, CSS3 et Tailwind CSS.

## Objectifs

L’objectif principal est donc de proposer une applications web qui répond à ces sept critères, qui sont ceux de l’entreprise, présent dans le cahier des charges.

## Respect du modèle MVC avec Laravel

Mettre en œuvre rigoureusement l’architecture Modèle-Vue-Contrôleur à chaque étape du développement, en séparant clairement la logique métier (modèles), la gestion des requêtes et règles (contrôleurs) et la présentation (vues Blade).

## Qualité et lisibilité du code

Produire un code clair et bien structuré, en respectant les conventions officielles Laravel, ainsi que les standards HTML5 et Tailwind CSS pour le front-end. Les noms de classes, méthodes et fichiers doivent être explicites et cohérents.

## Processus d’inscription et d’authentification complet

Implémenter l’ensemble des fonctionnalités utilisateurs : création de compte, connexion. L’ensemble doit être utilisable sans bug, avec des redirections.

## Sécurité des accès et des données

Appliquer les normes de sécurité Laravel : hachage des mots de passe, protection CSRF, gestion des autorisations via middleware et policies, et chiffrement des sessions. Les rôles et permissions doivent garantir que seuls les utilisateurs authentifiés accèdent aux pages protégées.

## Validation serveur et retours utilisateurs clairs

Mettre en place des règles de validation sur tous les formulaires côté serveur. En cas d’erreur, l’utilisateur reçoit un message précis indiquant le champ concerné et la nature du problème.

## Système de gestion des thés

Une interface complète de gestion des produits (CRUD) sera intégrée à l’application. Les utilisateurs pourront créer, modifier, supprimer ou consulter les informations liées aux thés (nom, variété, type, provenance, prix, etc.).

## Listes et exportation PDF

Les utilisateurs auront la possibilité de créer des listes de thés, en fonction de leurs préférences. Une fonctionnalité d’export au format PDF permettra de générer et sauvegarder ces listes de manière pratique.

## Planification

Ce projet a débuté le mercredi 30 avril 2025 à 8h00 et se terminera le lundi 26 mai 2025 à 11h25. Pour planifier ses différentes étapes, j’ai utilisé un modèle Excel fourni par l’ETML et conçu par Monsieur Lymberis.

Ce fichier, tout comme mon journal de travail, se présente sous la forme d’un tableau. Dans la première colonne, on peut sélectionner la catégorie de la tâche à réaliser, puis indiquer dans la suivante le nombre de périodes de 5 minutes que la tâche a nécessité. La colonne ‘’Explications’’ sert à décrire plus précisément ce qu’il est prévu de faire, et enfin une dernière colonne permet d’ajouter des liens de référence utiles.

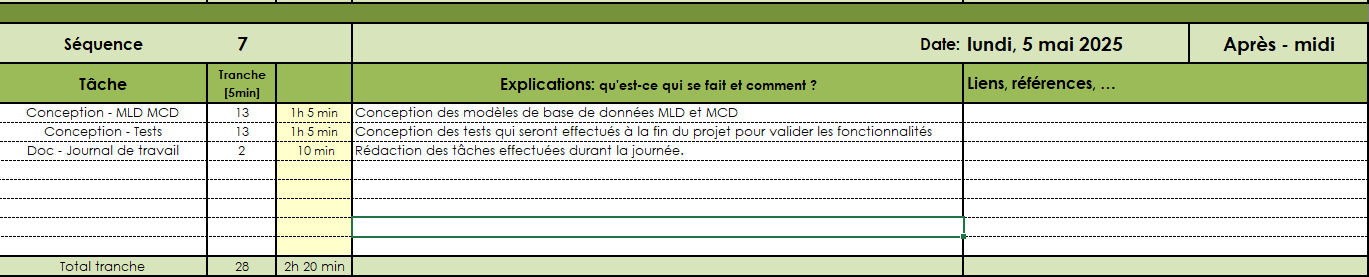
Voici un extrait illustrant une séquence planifiée :  


Figure 1 - exemple - Séquence de planification

Une fois toutes les tâches planifiées (comme ci-dessus), le fichier permet de générer un diagramme de Gantt qui illustre la planification et le suivi de l’avancement. Chaque barre verte représente la période prévue pour une tâche : on voit ainsi d’un coup d’œil ce qui était programmé et à quel moment du projet. Les barres bleues, quant à elles, proviennent du journal de travail et indiquent le temps réellement investi : elles permettent de comparer précisément la planification initiale à la réalité du terrain.

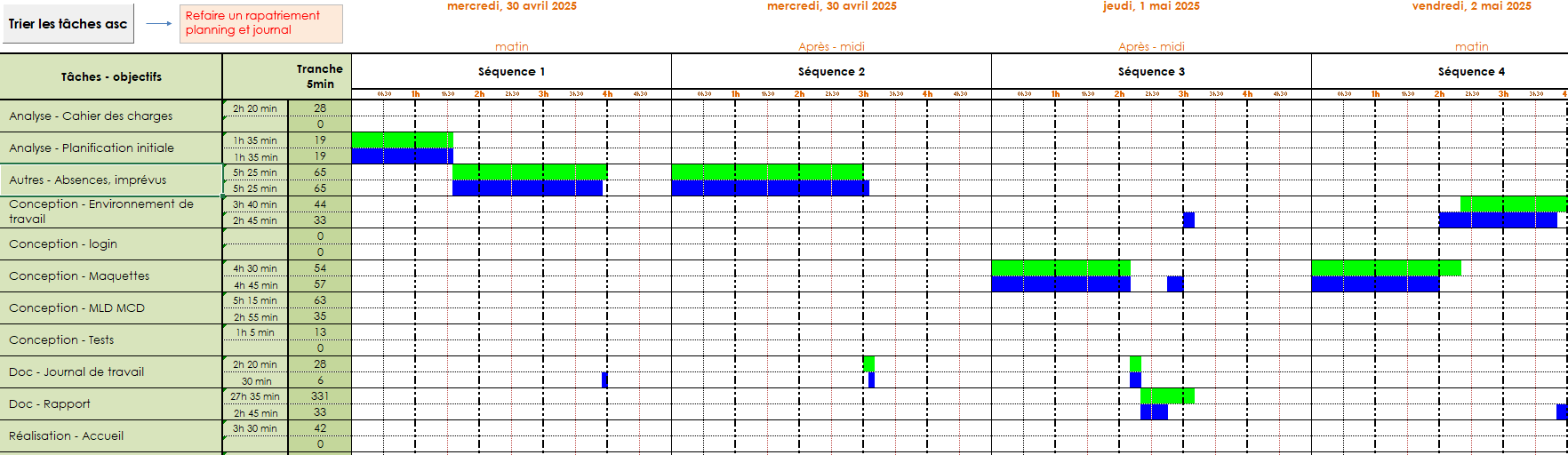
Voici le début de ce diagramme :  


Figure 2 exemple - Diagramme Gantt

*Révision de la planification initiale du projet :*

* *planning indiquant les dates de début et de fin du projet ainsi que le découpage connu des diverses phases.*

*Il s’agit en principe de la planification définitive du projet. Elle peut être ensuite affinée (découpage des tâches). Si les délais doivent être ensuite modifiés, le responsable de projet doit être avisé, et les raisons doivent être expliquées dans l’historique.*

## Planification initiale

Premièrement voici à quoi ressemble ma planification initiale sous forme de graphique :

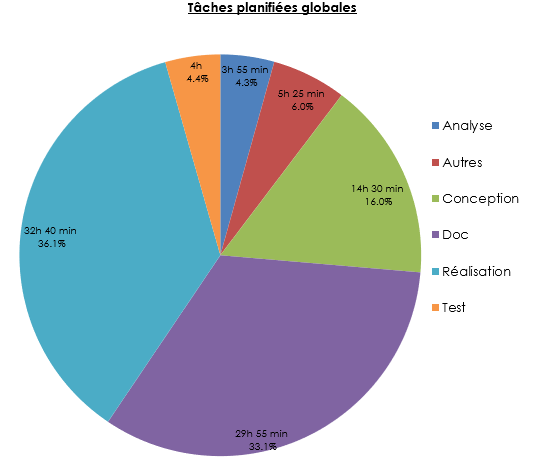


Figure 3 - graphique des tâches planifiées

On observe tout d’abord qu’avec près de 33 % du temps consacré à la documentation, celle-ci occupe une place majeure dans le projet, soulignant l’importance accordée à la traçabilité et à la qualité du livrable écrit.

L’analyse (4,3 %) et la conception (16 %) totalisent environ 20 %, exactement ce qui était préconisé dans le cahier des charges pour la phase d’étude.

La réalisation atteint 36 %, très proche des 35 % réservés à l’implémentation, ce qui confirme un bon respect de la répartition initiale.

Dans la catégorie ‘’autres’’ nous pourrons retrouver notamment les absences et imprévus potentiels, ces deux choses n’étant pas planifiables ces 6 % représente une légère marge de sécurité, du temps prévu en plus dans le cas où une fonctionnalité ne seraient pas terminé ou que la documentation devrait être finalisée.

Enfin, les tests représentent 4,4 % du planning, ce temps dédié à la validation des fonctionnalités garantit la fiabilité du logiciel.

## Méthodologie de travail

Pour ce projet, mon choix de méthodologie s’est tourné vers la méthode des six pas, celle-ci répond particulièrement bien aux exigences d’un TPI sur le développement web. Elle permet aussi une approche structurée avec une phase d’analyse approfondie qui permet de bien cerner les points importants et les objectifs du travail. La phase suivante se concrétise par la planification claire des tâches, suivie d’une conception méthodique des solutions. Le développement lui se fait de manière itérative, ce qui donne l’avantage de la flexibilité du projet en cours de route. Enfin, des phases de contrôle qualité rigoureuses assurent la fiabilité du produit final. Contrairement au modèle en cascade, cette méthode permet une progression plus souple, chaque étape pouvant évoluer indépendamment. Les dernières phases garantissent une évaluation complète avant la livraison, identifiant les points d'amélioration pour les projets futurs.

#### Décomposition en tâches selon les six pas

##### **S’informer**

* + Lecture et compréhension du cahier des charges pour cadrer l’application ExceptionTea.
  + Entretiens avec l’expert ou l’enseignant pour valider les objectifs.
  + Lecture d’informations sur Laravel, MySQL et Tailwind CSS.

##### **Planifier**

* + Découpage du projet en jalons (maquettes, prototype, développement, tests, documentation).
  + Estimation des durées pour chaque phase et chaque tâche.
  + Mise en place du dépôt GitHub et configuration du suivi de version.

##### **Décider**

* + Validation des wireframes basse fidélité pour chaque page clé.
  + Choix des librairies front-end (Tailwind CSS, plugin Vite) et des outils de génération de PDF (Dompdf ou Snappy).
  + Élaboration du MCD, du MLD et du MPD pour modéliser la structure de la base de données.
  + Conception des maquettes des différentes pages.

##### **Réaliser**

* + Développement back-end : création des modèles, migrations, contrôleurs et routes.
  + Développement front-end : intégration des maquettes en Blade, application des styles Tailwind et interactivité (pop-ups, filtres).
  + Configuration de la base de données MySQL et exécution des migrations Laravel.
  + Rédaction de la documentation technique et du journal de bord.

##### **Contrôler**

* + Exécution des tests unitaires et fonctionnels (vérification des flux CRUD, sécurité, validation des formulaires).
  + Revue de code et corrections éventuelles (refactoring au normes, ajustement des performances).
  + Préparation de la démonstration finale et génération des livrables.

##### **Evaluer**

* + Recueil des retours venant de l’évaluation du chef de projet et des experts.
  + Analyse des écarts entre le cahier des charges et le résultat livré.
  + Identification des améliorations et des bonnes pratiques à retenir pour un futur projet.

# Analyse / Conception

## Maquettes

J’ai choisi de réaliser des maquettes plus haute-fidélités au lieu de simples wireframes, afin de faciliter la phase frontend et d’éviter toute hésitation lors de l’intégration.

Les voici :



Figure 4 - maquette - S'enregistrer

Cette page permet de s’enregistrer la première fois, lorsqu’on n’a pas encore de compte.  
Pour ce faire, il faut entrer son nom, son adresse e-mail et un mot de passe. Puis la validation se fait avec le bouton ‘’S’enregistrer’’.

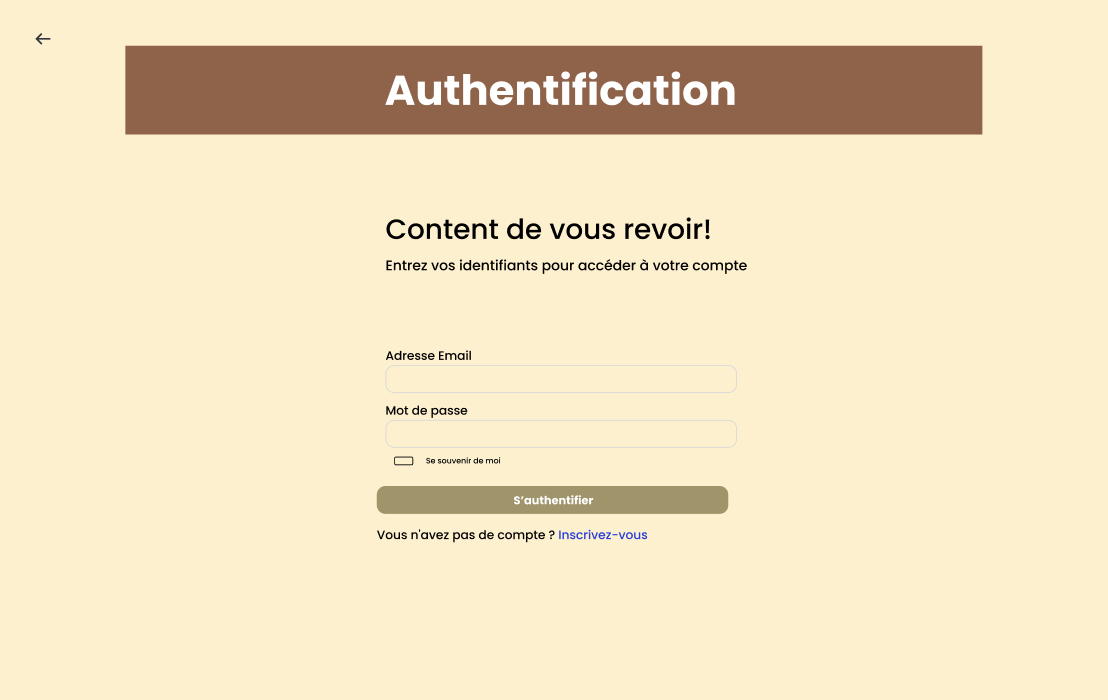
**

Figure 5 - maquette - S’authentifier

Ici nous avons la page qui permet de se connecter si on a déjà un compte, il suffit de rentrer son adresse mail et son mot de passe dans les champs prévus à cet effet. Puis la validation se fait avec le bouton ‘’S’authentifier’’.

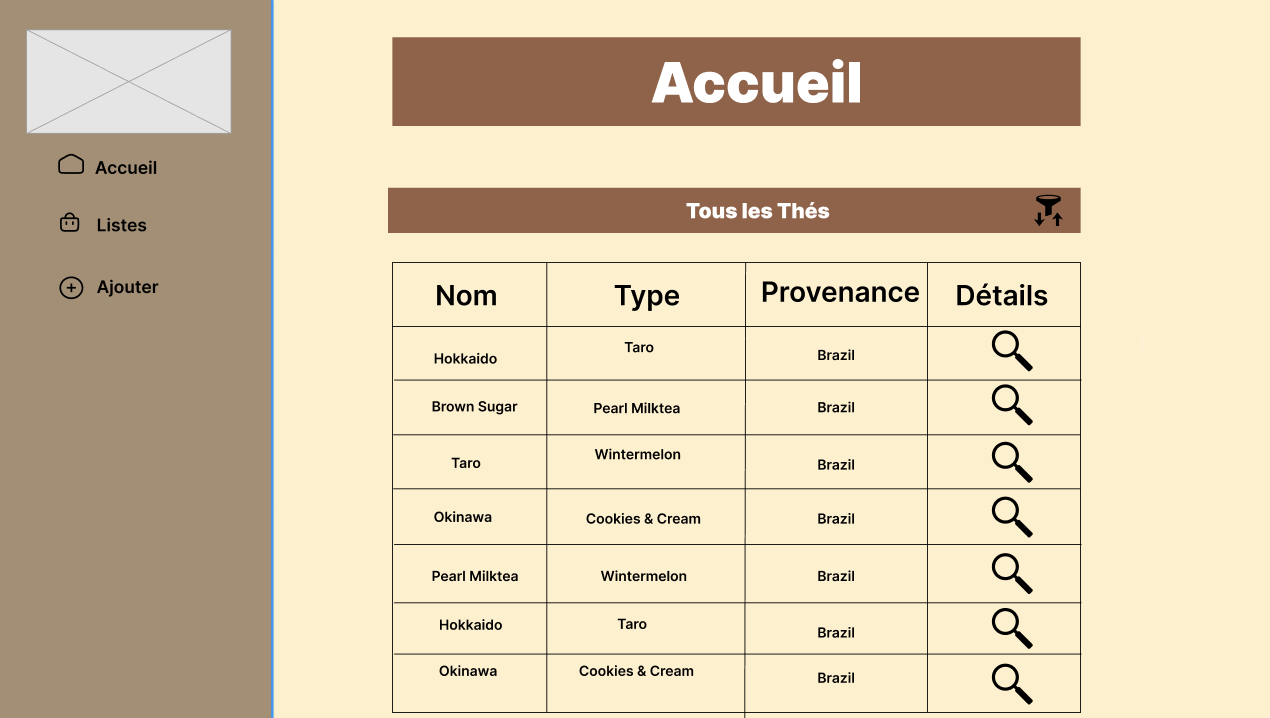


Figure 6 - maquette - Accueil

Voici la page d’accueil. On y trouve une liste de thés disponibles dans la boutique, avec des options de tri et de filtrage. Il est possible d’accéder à la page de détails d’un thé en cliquant sur l’icône en forme de loupe située à droite de chaque élément. À gauche, des boutons de navigation permettent de parcourir les différentes pages du site.

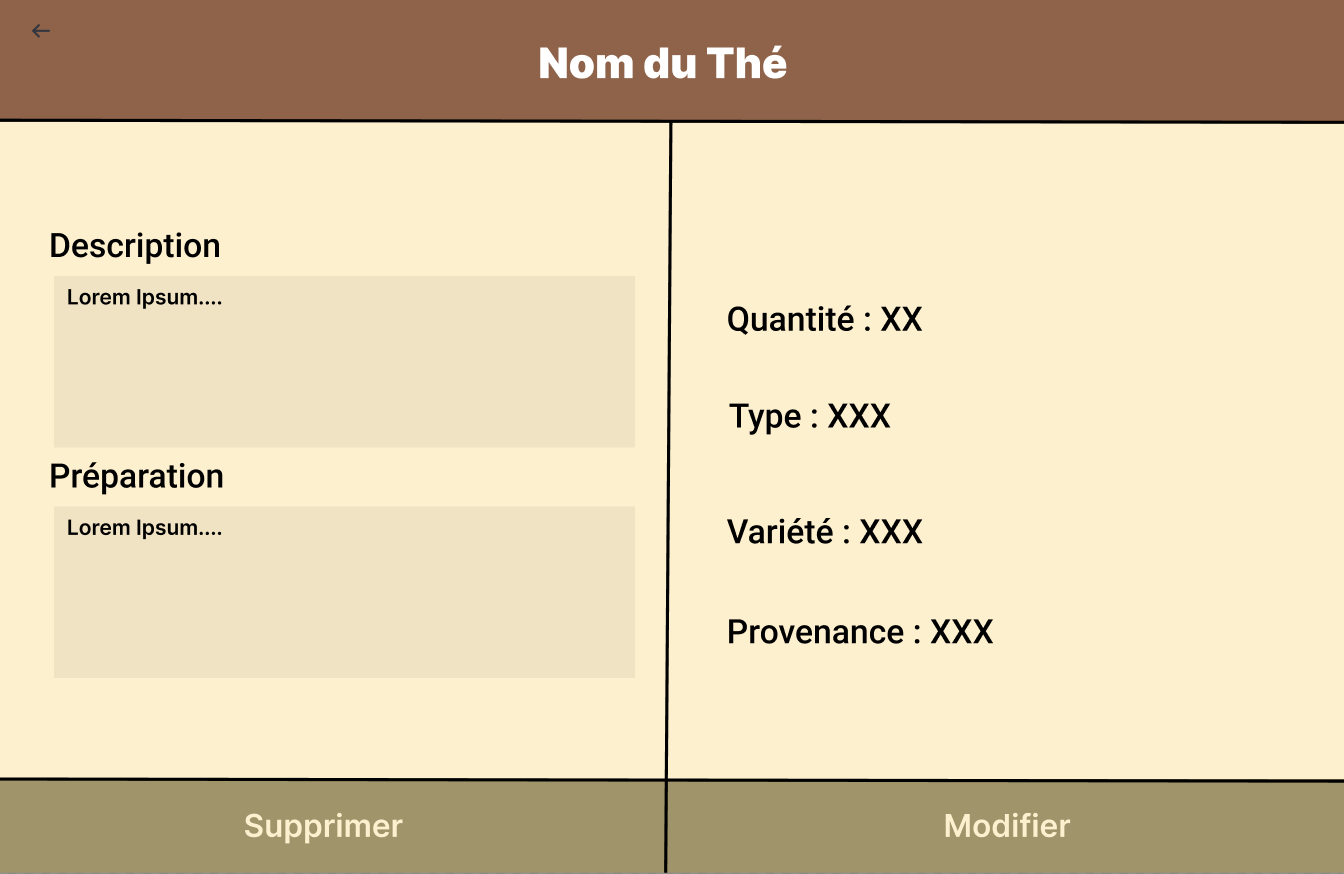
**

Figure 7 - maquette - Détails du Thé

Ici, la page qui permet d’obtenir plus d’informations sur le thé sélectionné. On peut y consulter des éléments tels que la description, les instructions de préparation, la quantité en stock, la variété et la provenance du thé. Deux boutons situés dans le pied de page permettent de modifier ou de supprimer les informations liées à ce thé.



Figure 8 - maquette - Ajouter un thé

Sur cette page, nous avons un formulaire permettant d’ajouter un nouveau thé dans la base de données. Nous devons renseigner toutes ces informations importantes dans des champs prévus à cet effet. Les cinq premiers champs sont essentiellement des champs texte. Le champ de la date aura une icône de calendrier à droite et sera de type ‘’date’’, ce qui facilite la sélection. Les trois dernières informations à renseigner sont la variété, la provenance et le type. Ces trois champs sont sous forme de listes déroulantes, il suffit de sélectionner l’information correcte dans la liste pour l’assigner au nouveau thé en question. Ces listes déroulantes permettent également de supprimer des options grâce à la croix à droite de l’élément, de modifier des éléments avec l’icône de crayon et d’ajouter un élément avec le bouton "Ajouter un élément" en fin de liste.

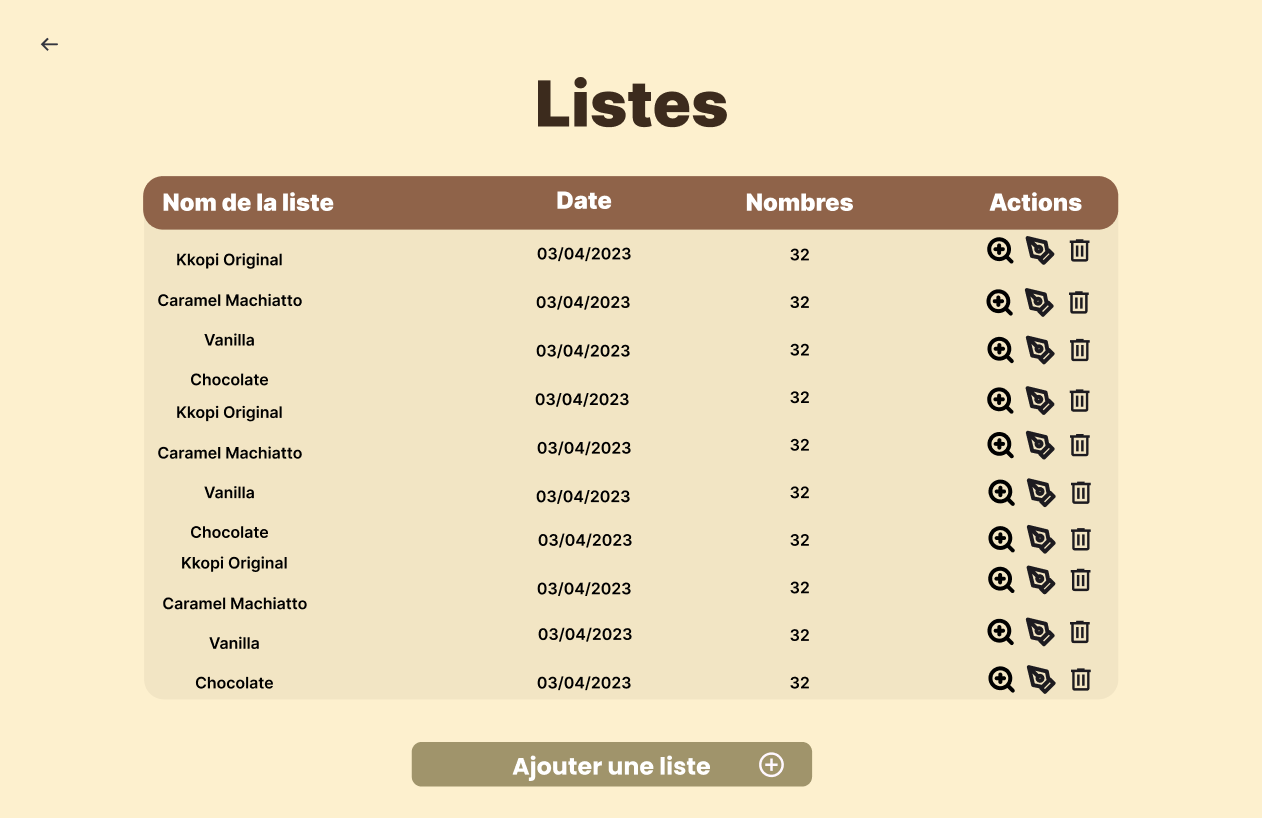


Figure 9 - maquette – Listes

Ci-dessus, la page contenant toutes les listes de thés disponibles, répertoriées par leur nom, la date de création et le nombre de thés présents dans chaque liste. Dans la colonne "Actions", nous avons trois boutons :

1. La loupe qui affiche les détails de la liste,
2. Le stylo qui modifie les éléments de la liste,
3. La poubelle qui supprime entièrement la liste.

**

Figure 10 - maquette - Détails d'une liste

Voici la page de détails d’une liste, elle permet à l’utilisateur de voir le contenu de toute la liste ainsi que de l’exporter en PDF avec le bouton vert.

## Modélisation de la base de données

## MCD

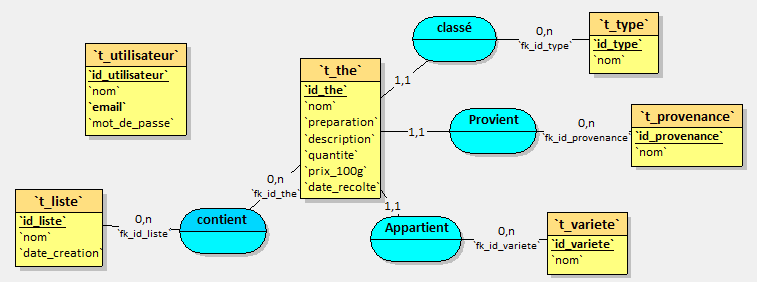


Figure 11 - BDD – MCD

Le MCD présente, de manière visuelle et abstraite, les **entités** clés du domaine, ici Thé, Provenance, Type, Variété, Utilisateur et Liste en plus de leurs **relations**. Chaque entité est représentée par un rectangle listant ses principales propriétés (nom, description, etc.), et chaque association par un ovale verbal (par exemple ‘’contient’’ pour relier Liste et Thé, ‘’Provient’’ pour relier Thé et Provenance). Ce schéma garantit que tous les éléments métier et leurs interactions sont correctement identifiés avant toute implémentation technique.

## MLD

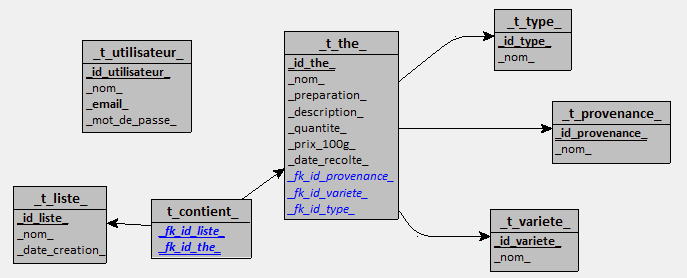
******

Figure 12 - BDD – MLD

Le MLD correspond à une version technique du MCD, mieux adaptée aux contraintes d’un système de gestion de base de données relationnelle comme MySQL. Il formalise chaque entité du MCD en une table concrète avec des noms et une clé primaire. Les associations deviennent des liens explicites via des clés étrangères (fk\_...), intégrées aux tables concernées. La relation plusieurs(0,n)-à-plusieurs(0,n) entre les listes et les thés est gérée par une table d’association appelée t\_contient, dont la clé primaire composite (fk\_id\_liste, fk\_id\_the) permet d’éviter les doublons dans les liens.

## MPD

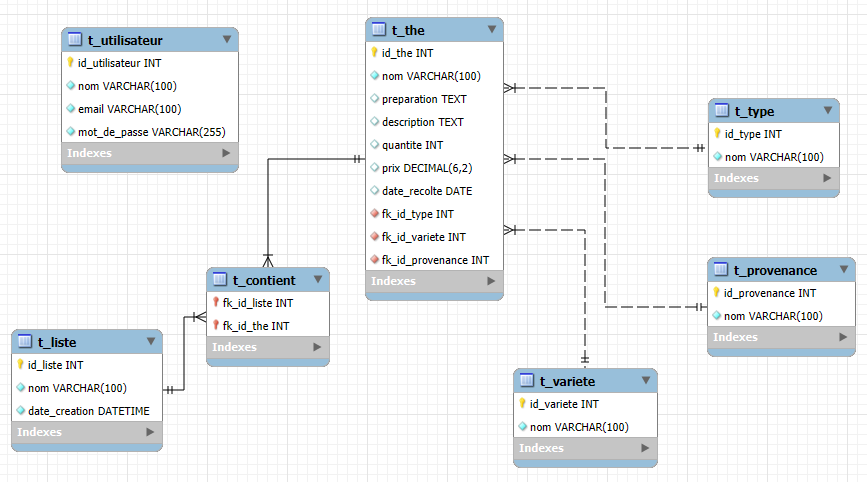


Figure 13 - BDD – MPD

Le MPD correspond à la représentation concrète de la base de données directement issue de l’outil Workbench de MySQL. Il affiche chaque table avec ses colonnes, types de données, clés primaires et clés étrangères, ainsi que les connexions schématisées entre elles. Ce schéma reflète exactement ce que le SGBD met en œuvre, prêt à être en production.

## Stratégie de test

La stratégie de tests vise à garantir que l’application répond précisément aux exigences fonctionnelles et non fonctionnelles définies dans le cahier des charges, tout en assurant la robustesse et la maintenabilité du code. Elle se déploie selon deux axes :

1. Tests fonctionnels (Feature Tests)
2. Tests unitaires

## Tests fonctionnels

Objectif : vérifier que chaque fonctionnalité clé du cahier des charges fonctionne dans son ensemble.

Exemples de scénarios :

1. Inscription / connexion : un nouvel utilisateur peut créer un compte et se connecter.
2. CRUD sur les thés : ajout, modification, consultation et suppression d’un thé.
3. Listes de thés : création d’une liste, ajout et retrait de thés, export PDF.

Exécution : le but est de rédiger d’abord une fiche simple par scénario (étapes et résultat attendu).  
Une fois la fonctionnalité implémentée, on reprend cette fiche pour valider les tests, à condition qu’ils respectent entièrement le scénario décrit.

Voici un tableau des tests fonctionnels :

|  |  |  |  |
| --- | --- | --- | --- |
| **N° du test** | **Fonctionnalité** | **Action** | **Résultat attendu** |
| 1 | Inscription / authentification | Tenter de s’inscrire avec un nouvel e-mail valide puis se déconnecter et se reconnecter | L’inscription crée un compte, la connexion fonctionne et l’utilisateur est redirigé vers le dashboard |
| 2 | Sécurité utilisateur (mot de passe, autorisations) | Tenter d’accéder à /the/create sans être connecté | L’accès est refusé |
| 3 | Validation serveur des formulaires | Soumettre le formulaire d’ajout de thé avec un champ « prix » vide | Le formulaire est rejeté, et des messages d’erreur clairs s’affichent sous chaque champ concerné |
| 4 | Gestion du stock de produits (CRUD) | Créer un nouveau thé, puis modifier sa quantité, le consulter et enfin le supprimer | Après chaque action, les données sont correctement enregistrées / affichées / supprimées dans la base MySQL |
| 5 | Listes personnalisées + export PDF | Créer une liste de thés, y ajouter plusieurs thés, puis cliquer sur « Exporter en PDF » | Un fichier PDF est téléchargé, contenant la liste des thés avec leur nom. |

Le respect du modèle MVC ainsi que la qualité du code (conventions Laravel, Tailwind CSS, HTML5) ne faisant pas partie des tests fonctionnels, ils ne figurent donc pas dans le tableau ci-dessus.

## Tests unitaires

Objectif : vérifier que les fonctions essentielles du code donnent les bons résultats, sans dépendre de l’interface ou de la base de données.

Exemples de cibles :

1. Methods de validation des modèles (par exemple, règles de quantité et de prix).
2. Calculs éventuels (par exemple, conversion de dates ou formats).

Outil : Unit Tests de Laravel :

php artisan make:test … --unit

Exécution : peut se faire avec la commande :

php artisan test --filter Unit

## Données de test

Les tests ont été effectués avec des données simulées, pas de données réelles pour des raisons de sécurité.

## Risques techniques

Plusieurs points de vigilance ont été identifiés :

## Apprentissage sur bibliothèques externes

L’intégration de Dompdf par exemple ou d’un outil équivalent pour l’export en PDF peut demander un peu de temps de prise en main supplémentaire. Pour réduire cet impact, je consulterai rapidement une documentation officielle ou des vidéos sur le sujet afin de maîtriser les fonctionnalités essentielles sans retarder mon planning.

## Expérience limitée en JavaScript

Le projet requiert des interactions dynamiques (pop-ups, filtres, …). Comme JavaScript n’a pas été vu en cours, il est possible que certaines mises en œuvre prennent plus de temps que prévu. Pour y faire face, j’envisage de réutiliser des exemples de scripts simples et de privilégier des snippets éprouvés, plutôt que d’apprendre un framework au complet.

## Familiarité restreinte avec Laravel

Bien que Laravel soit réputé pour son intuitivité, je ne l’ai utilisé qu’au cours d’un projet pré-TPI. Certaines conventions ou fonctionnalités avancées pourraient nécessiter une phase de repérage. Pour limiter ce risque, je m’appuierai sur la documentation officielle et les exemples issus de mon projet de pré-TPI.

## Dossier de conception

Le dossier de conception décrit l’environnement de travail mis en place pour le projet, ainsi que les outils principaux utilisés. Il sert à poser le cadre technique avant de coder réellement.

## Environnement

Ce projet se réalise intégralement en local, sur un poste de travail standard (Windows 10 ou équivalent) disposant d’une configuration matérielle suffisante pour exécuter un IDE et un navigateur moderne sans latence. Aucune infrastructure serveur externe n’est requise durant la phase de développement : tout tourne directement en local.

## Outils

Voici les outils utilisés durant ce projet ainsi que leur version :

|  |  |
| --- | --- |
| **Outils** | **Versions** |
| PHP | 8.3.4 |
| Laravel | 12.12.0 |
| TailwindCSS | 4.1.5 |
| MySQL | 8.4.5 |
| MySQL Workbench | 8.0.42 |
| Looping | 4.1 |
| Composer | 2.8.8 |
| NodeJS | 22.15.0 |
| NPM | 10.9.2 |
| Visual Studio Code | 1.99.3 |
| GitHub Desktop | 3.4.19 |
| Vite | 6.3.4 |
| Breeze | 2.3.6 |
| Blade | 12.12.0 |
| Pest | 3.8.2 |

Les descriptions de ces outils se trouvent toutes dans [le glossaire.](#_Glossaire)

# Réalisation

## Problèmes rencontrés

Au cours du développement d’ExceptionTea, plusieurs erreurs ou problèmes sont apparus. Parcourons dans cette partie les plus marquantes d’entre-elles et comment les résoudre.

## Mode ‘’safe update’’ de MySQL

#### Problème

Problème : l’exécution d’un script ‘’DELETE … JOIN’’pour supprimer les doublons a été bloquée par l’erreur

Error Code: 1175. You are using safe update mode…

#### Cause

MySQL Workbench était en “safe update mode”, interdisant les DELETE sans clause WHERE sur une clé indexée.

#### Solution

Au début du script SQL, désactiver temporairement le mode sécurisé :

SET SQL\_SAFE\_UPDATES = 0;

Ensuite exécuter nos suppressions de doublons.

Rétablir le mode sécurisé :

SET SQL\_SAFE\_UPDATES = 1;

Ou bien décocher l’option “Safe Updates” dans les préférences Workbench, puis se reconnecter.

## Timeout de processus

#### Problème

Erreur "The process exceeded the timeout of 60 seconds" lors de l'exécution de tâches dans la file d'attente.

#### Cause

Des processus trop longs dans le traitement des tâches de la file d'attente.

#### Solution

Augmenter la limite de timeout ou optimiser les processus exécutés dans la file d'attente

## Menu latéral statique

#### Problème

Sur la page d’ajout de thé, la barre de navigation à gauche ne s’étendait pas jusqu’en bas lorsque l’on ouvrait une des listes déroulantes servant à sélectionner un Type, une variété ou une provenance. Le bouton ‘’se déconnecter’’ restait à hauteur fixe et la partie gauche de la page, normalement réservée à la navbar, était de la même couleur que le fond du site.

#### Cause

La classe CSS ‘’h-screen’’ (hauteur écran) fixait rigoureusement la hauteur du conteneur, sans tenir compte du padding et de l’expansion induite par le menu déroulant. De plus, la propriété overflow-hidden empêchait tout défilement.

#### Solution

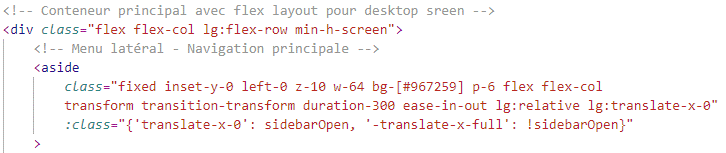
Sur l’extrait suivant, on utilise une ‘’div’’ à la racine qui passe automatiquement d’une colonne en mobile à une ligne en bureau grâce aux classes ‘’flex flex-col lg:flex-row min-h-screen’’. Cela signifie que, quel que soit l’écran, le conteneur occupe toujours la hauteur complète de la fenêtre.

Figure 14 - solution - navbar – code

Puis à l’intérieur, la balise ‘’<aside>’’ définit la barre de navigation latérale : elle est positionnée en fixe (fixed inset-y-0 left-0) sur mobile et peut se glisser hors écran ou revenir (translate-x-full / translate-x-0) via les utilitaires Tailwind transform transition-transform duration-300 ease-in-out. Dès que la taille atteint le point lg, on passe en mode relatif (lg:relative lg:translate-x-0), ce qui maintient la sidebar toujours visible. Cette combinaison de ‘’flexbox’’, de positionnements et de transitions nous assure que la navigation reste accessible, ne masque jamais son contenu, et s’adapte dynamiquement même si un sous-menu se déploie.

## Fonctionnalités importantes

Voici une liste des fonctionnalités clés de ce projet avec leur explications détaillée.

## Sécurité des données

Voyons comment nous gérons la sécurité des données et des accès dans ce projet.

#### Hachage des mots de passe

Laravel utilise par défaut l’algorithme ‘’**Bcrypt’’** pour hacher les mots de passe. Lorsque l’on crée un nouvel utilisateur, la méthode ‘’Hash::make($password)’’ transforme la chaîne en un haché sécurisé. Cette configuration est définie dans le fichier ‘’config/hashing.php’’.

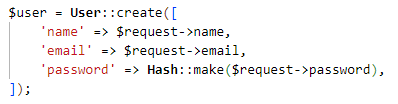
Et voici à quoi ressemble cette fonction :   


Figure 15 - sécurité - hash

#### Protection CSRF

Laravel génère automatiquement un jeton CSRF (Cross-Site Request Forgery) unique pour chaque session. Il suffit d’ajouter la directive Blade ‘’@csrf’’ dans chaque formulaire, ce qui insère un champ caché contenant ce jeton.

Voici un exemple :

<input type="hidden" name="\_token" value="{{ csrf\_token() }}">

Le middleware ‘’VerifyCsrfToken’’, activé par défaut, vérifie à chaque requête d’état (POST, PUT, DELETE) que le jeton reçu correspond à celui stocké en session. En cas d’absence ou d’invalidité, la requête est rejetée (erreur 419).

#### Chiffrement des sessions

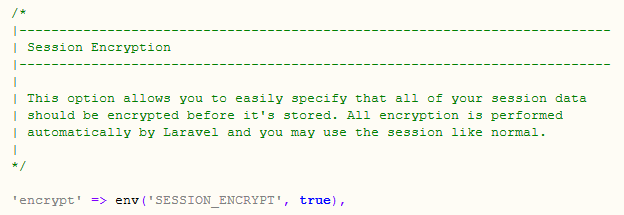
Toutes les données de session sont chiffrées avec la clé d’application définie dans ‘’.env’’ (APP\_KEY). Le driver de session est configuré dans ‘’config/session.php’’ :  


Figure 16 - sécurité – Chiffrement

Grâce à cela, même si un attaquant accédait aux fichiers de session (ou à la table de sessions), il ne pourrait pas lire les informations sans la clé.

#### Middleware & policies

Dans Laravel, on peut gérer l’accès aux pages destinées aux utilisateurs authentifiés grâce à un middleware auth.

Concrètement, les routes publiques (inscription, connexion, mot de passe oublié) sont définies sans middleware.

Les routes qui suivent l’authentification sont placées dans un **groupe** auquel on applique ->middleware('auth'). Par exemple :

**Comportement en cas d’accès non autorisé :**

Si un utilisateur non connecté tente d’accéder à une route protégée, Laravel détecte l’absence de session valide.

**Injectée par défaut** dans le groupe web, la clé auth est gérée par la classe App\Http\Middleware\Authenticate.

Cette classe redirige automatiquement vers la page de connexion, garantissant que seules les personnes authentifiées peuvent atteindre les pages internes.

**Avantage**

Centralisation : aucun test manuel de session à écrire dans chaque contrôleur.

Simplicité : il suffit d’ajouter ->middleware('auth') ou d’inscrire la route dans le groupe pour bénéficier de la protection.

## Barre de Recherche, filtres et tri

#### Visuel

##### Barre de recherche

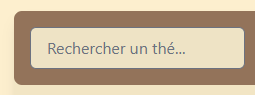
Sur la page d’accueil (listing de tous les thés), une barre de recherche textuelle a été ajoutée pour faciliter la navigation lorsque la liste devient longue.  
Elle se comporte en filtrant en temps réel la liste des thés dont le nom contient la chaîne tapée (recherche insensible à la casse).  
Voici son visuel :   


Figure 17 - barre de recherche

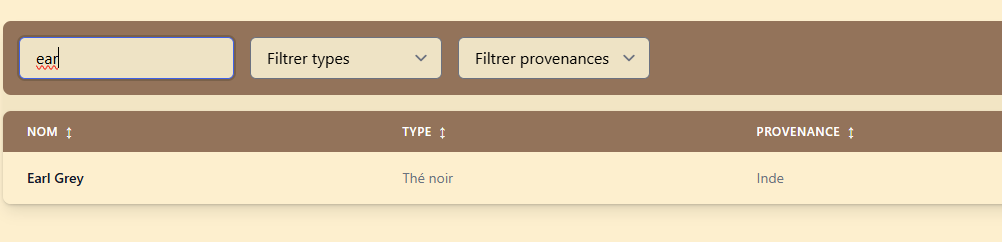
Par exemple si on cherche ‘’ear’’ il nous filtre la liste et n’affiche que le thé dont le nom commence par ces trois lettres :  


Figure 18 - affichage - recherche

Si nous avions d’autres thé commençant par les mêmes caractères, ils apparaîtraient aussi à la suite.

##### Filtrage

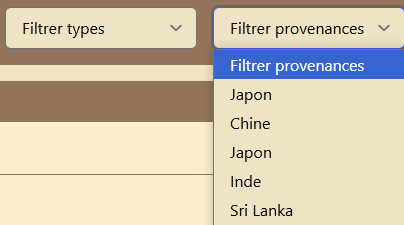
Il y a deux filtres disponibles, le filtrage par type de thé et celui par leur provenance.  
Voici à quoi ressemble les deux listes.  


Figure 19 - visuel - filtrage

Pour sélectionner le type ou la provenance que l’on souhaite nous devons cliquer sur l’icône de flèche à droite et une liste déroulante s’ouvre (comme sur le filtre de droite).

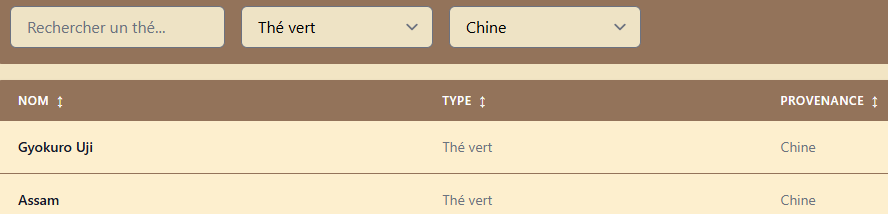
Nous pouvons choisir de cumuler deux filtres puis seuls les éléments qui contiennent les deux critères apparaissent. Comme dans cet exemple :   


Figure 20 - exemple - filtrage

##### Tri

Chaque en-tête de colonne (Nom, Type, Provenance) affiche une double flèche « ↕ »  
Un premier clic trie la colonne par ordre alphabétique.  


Figure 21 - exemple - tri

Un second clic sur cette double flèche inverse l’ordre.

#### Fonctionnement

À présent nous allons voir comment se traduit ces fonctionnalités dans le code :

##### Tri

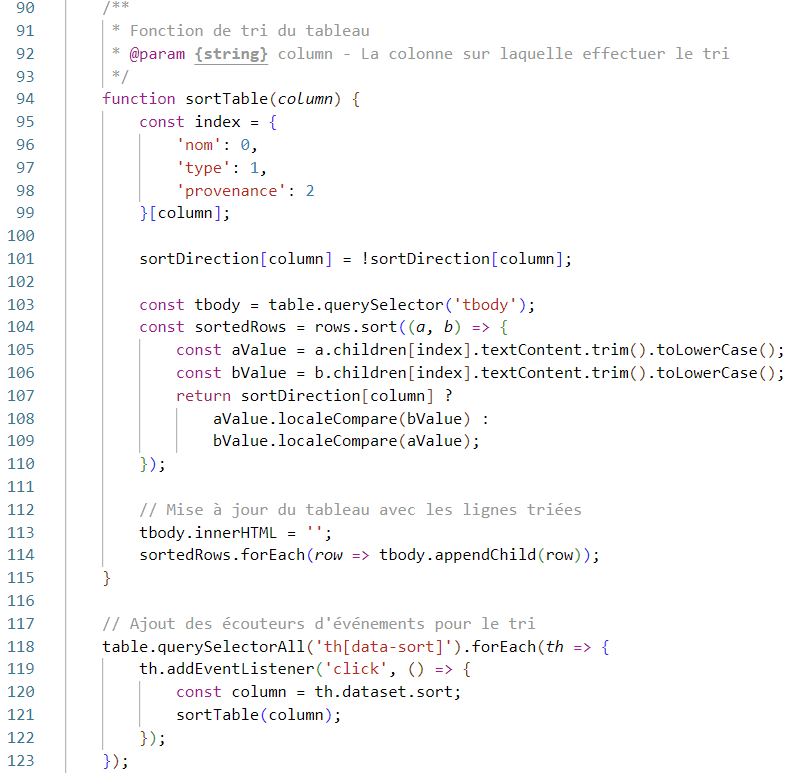
La partie code du tri, s’effectue de la ligne 90 à la 123 dans la fonction ‘’sortTable’’.  


Figure 22 - code - fonction – sortTable

Lignes 95 à 99 on détermine quel index de cellule trier selon la colonne.

Ligne 101 : on bascule le sens du tri (asc/desc).

Lignes 103–110 : on trie le tableau de lignes en mémoire, puis on réinjecte l’ordre dans le ‘’<tbody>’’.

Lignes 118–121 : on associe la fonction aux clics sur les en-têtes marqués ‘’data-sort’’.

##### Filtrage et recherche

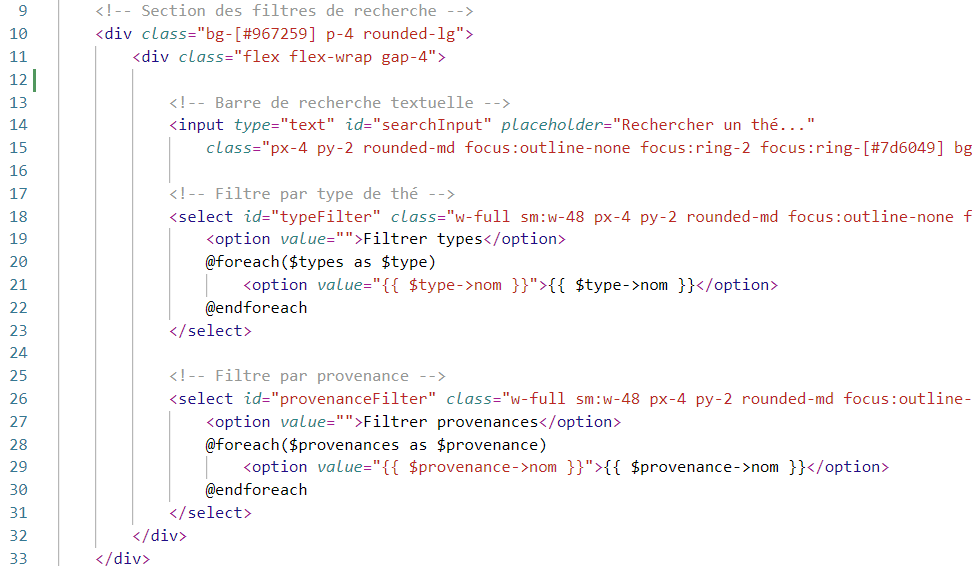
Dans notre fichier de vue dashboard.php nous avons d’abord le code html qui permet de faire apparaitre ces options de filtrage des thés :   


Figure 23 - code – section filtrage de recherche - HTML

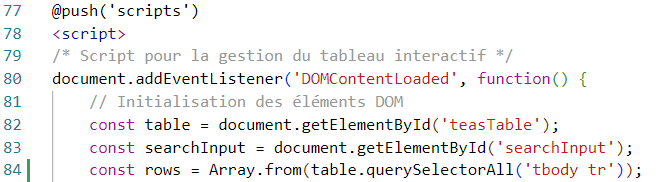
Nous pouvons voir que la barre recherche est un simple input de type texte avec un Id de la ligne 13 à la 15.  
Les lignes 17 à 23 sont celles pour la liste déroulante servant à choisir le type de thé à filtrer, avec une boucle foreach qui affiche tous les types disponibles.  
Pour la provenance au lignes 25 à 31 c’est sensiblement la même chose que le filtre de type.   
Voilà pour la partie HTML, maintenant regardons comment ces trois filtrages sont codés.  
  
En suivant l’id ‘’searchInput’’ dans le code, nous arrivons à la partie technique où le script et les méthodes sont déclarés. Ce script qui gère toute la partie tri et filtre :   


Figure 24 - code - début du script

Ici, la ligne 80 s’assure que le script ne s’exécute **qu’après** le chargement complet de l’HTML.  
La ligne 82 récupère le tableau ‘’teasTable’’ qui contient la liste de tous les thés disponibles, cet élément sera ensuite filtré.  
La ligne 83 sélectionne le champ de recherche textuelle ; c’est lui qui capte la saisie de l’utilisateur pour filtrer le tableau.  
Ligne 84 transforme la liste des balises ‘’<tr>’’ du ‘’<tbody>’’ en tableau JavaScript, pour pouvoir itérer facilement sur chaque ligne.

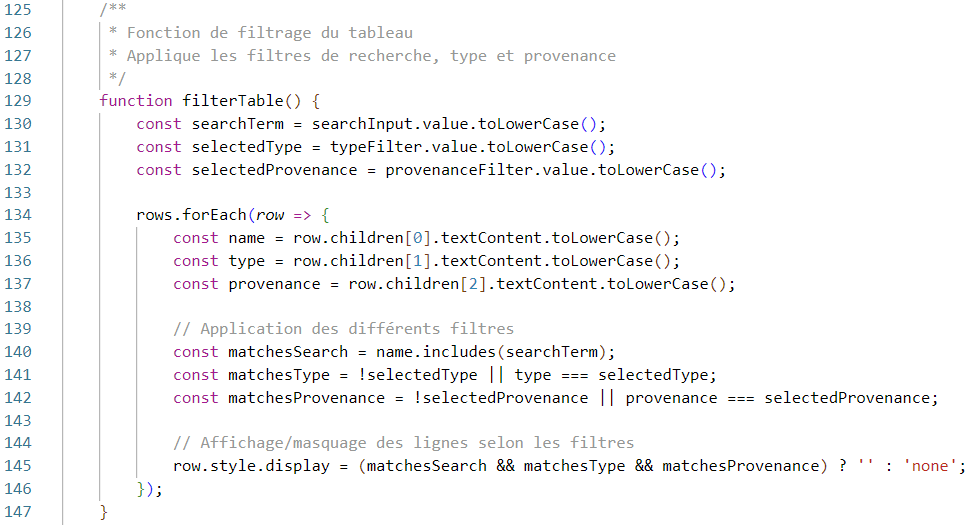
Viens notre première fonction :   


Figure 25 - code - fonction - filterTable

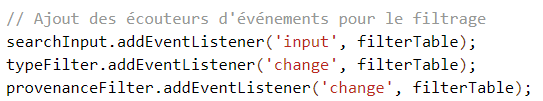
Ici nous avons donc la fonction filterTable qui applique tous les filtres en même temps à notre liste de thés.   
La ligne 130 récupère le texte entré dans la barre de recherche et le converti en minuscule pour le rendre insensible à la casse.   
Puis les deux lignes en dessous récupèrent la valeur du filtre type et du filtre provenance.   
Le forEach de la ligne 134 à 146 parcours, dans un premier temps (135-137), chaque ligne en extrayant le texte de chaque cellules (en LowerCase pour que la recherche soit insensible à la casse). Puis il teste les trois conditions sur les valeurs extraites aux lignes 140 à 142.  
  
Finalement la ligne 145 affiche la ligne dans le tableau si celle-ci respecte bien tous les critères.  
  
Il nous reste à attacher les filtres à un Eventlistener (écoute les évènements) pour qu’à chaque modification de texte dans la barre de recherche ou à chaque modification d’un filtre la fonction ‘’filterTable’’ soit réappelée. Comme ceci :  


Figure 26 - code - EventListener

## Connexion/authentification

Pour l’inscription et la connexion, pas besoin de tout coder à la main : c’est Laravel Breeze qui s’en charge.  
Ce plugin officiel contient par défaut :

* Les routes (/login, /register, /forgot-password, etc.)
* Les contrôleurs pour gérer l’authentification et la réinitialisation de mot de passe
* Les vues Blade, déjà stylées avec Tailwind CSS
* Le middleware auth et guest pour protéger les pages

Il suffit de lancer :

php artisan breeze:install

Puis :

npm install && npm run dev

Pour ensuite avoir un système d’authentification complet, sécurisé et prêt à l’emploi.

## Listes déroulantes de la page d’ajout

#### Visuel

Pour faciliter la catégorisation des thés, la page d'ajout propose trois listes déroulantes distinctes : Variété, Provenance et Type.   
Ces sélecteurs sont stylisés pour s'intégrer harmonieusement au design global de l'application.

Voici leur apparence sur la page :

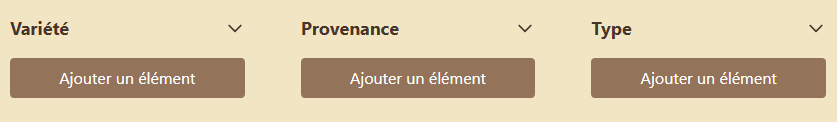


Figure 27 - visuel - listes déroulantes

Chaque liste présente les caractéristiques suivantes :

Un en-tête cliquable avec une icône de flèche indiquant qu'il s'agit d'un menu déroulant

Une zone de défilement lorsque la liste contient de nombreux éléments

Des boutons d'édition et de suppression à côté de chaque élément

Un bouton "Ajouter un élément" en bas pour créer de nouvelles entrées

Lorsqu'un utilisateur clique sur l'en-tête, la liste se déploie et affiche tous les éléments disponibles :



Figure 28 - visuel - listes déroulantes ouvertes

L'utilisateur peut sélectionner un élément en cliquant sur le bouton radio correspondant. Il est également possible de modifier ou supprimer un élément existant grâce aux icônes situées à droite.

Pour ajouter un nouvel élément, l'utilisateur clique sur le bouton "Ajouter un élément", ce qui ouvre une fenêtre de dialogue simple :



Figure 29 - visuel - pop-up ajout

#### Fonctionnement

Le fonctionnement technique de ces listes déroulantes repose sur plusieurs mécanismes JavaScript qui interagissent avec le backend via des appels API.

3.2.3.2.1 Structure HTML

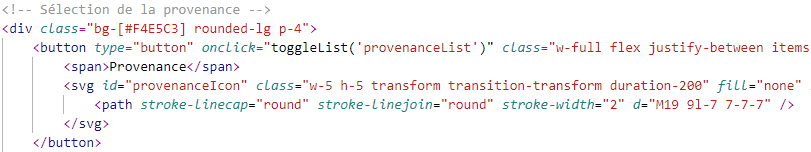
Chaque liste est structurée de manière similaire, comme on peut le voir dans ces extraits du code pour la liste des Provenances :  
En premier le bouton pour ouvrir la liste :   


Figure 30 - code - bouton - listes déroulantes

Puis ici un foreach qui nous affiche chaque provenance disponible avec deux boutons à sa droite, modifier (qui ouvre une pop-up) et supprimer.

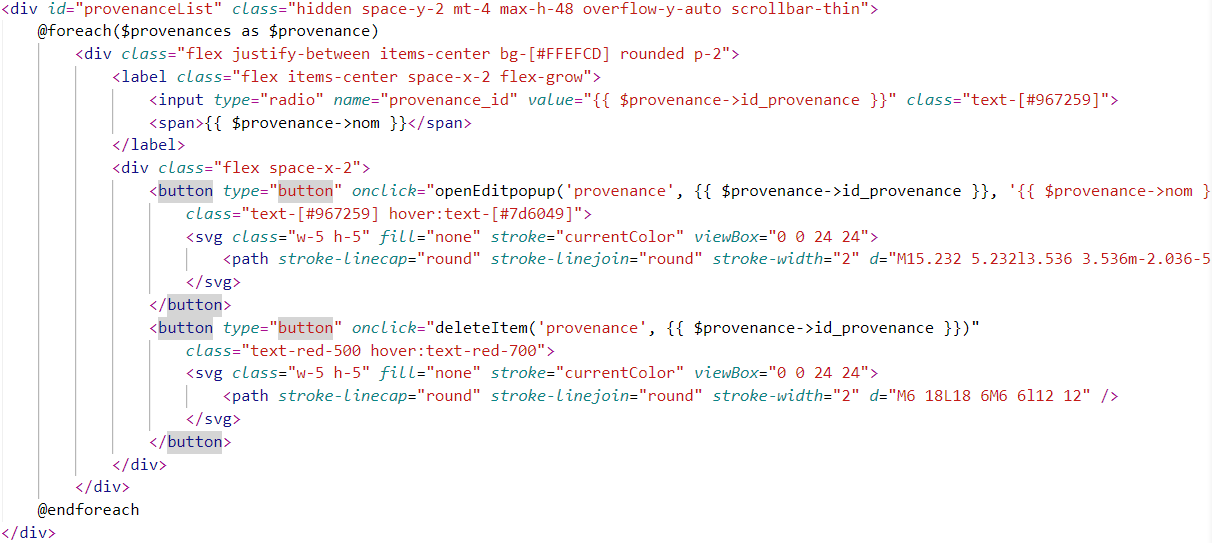


Figure 31 - code - liste

Les trois sections (Variété, Provenance, Type) utilisent le même modèle avec simplement des données différentes.

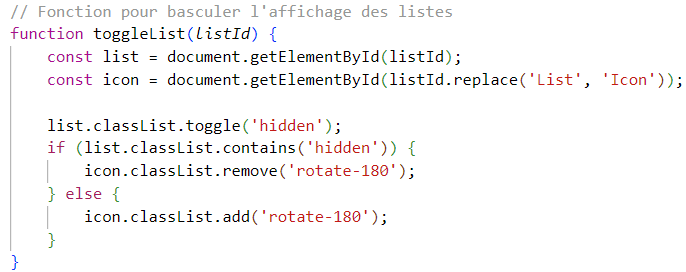
Pour la Gestion de l'ouverture/fermeture des listes déroulantes c’est la fonction ‘’toggleList’’qui s’en charge : 

Figure 32 -code - fonction - toggleList

Cette fonction alterne la visibilité de la liste et fait pivoter l'icône de flèche pour indiquer l'état actuel (ouvert ou fermé).

Lorsqu'un utilisateur souhaite ajouter ou modifier un élément, un popup apparaît grâce aux fonctions ‘’openAddpopup’’ et ‘’openEditpopup’’. Ces fonctions préparent le formulaire avec les bonnes valeurs et affichent la boîte de dialogue.

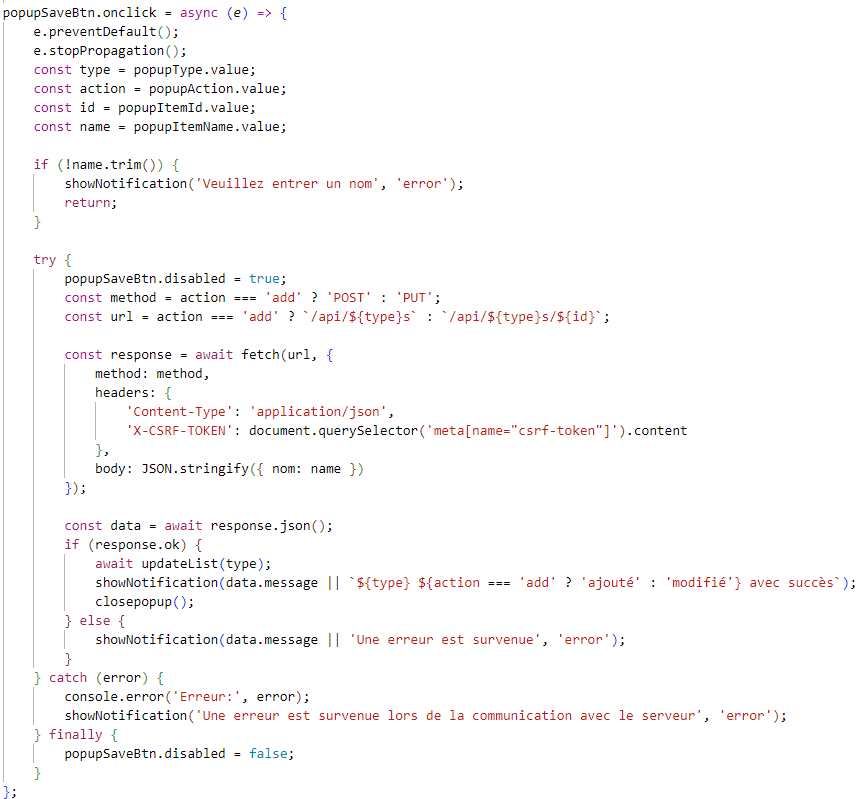
Le bouton de sauvegarde du popup utilise ensuite des requêtes AJAX pour communiquer avec l'API : 

Figure 33 - code - fonction - Pop-up

3.2.3.2.4 Mise à jour dynamique des listes

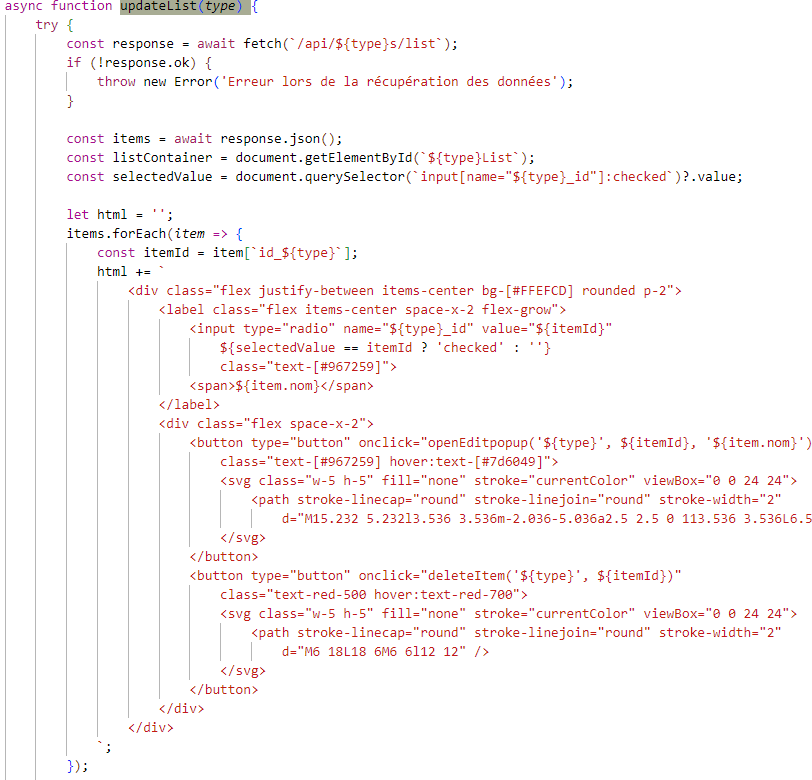
Après chaque opération, la fonction updateList est appelée pour rafraîchir le contenu de la liste concernée sans avoir à recharger toute la page :

Figure 34 - code - fonction - updateListe

Cette approche permet une expérience utilisateur fluide et réactive, sans les temps de chargement qu'impliquerait un rechargement complet de la page.

## Exportation en PDF

#### Visuel

La fonction d'export PDF permet aux utilisateurs de générer un document PDF à partir de la page de détails d’une liste de thés, et ce, grâce à un bouton ‘’Exporter en PDF’’ que voici :



Figure 35 - visuel - Exporter PDF

Ce document est ensuite téléchargé automatiquement puis peut être imprimé, partagé etc....

Le PDF généré présente un format clair avec :

* Un en-tête contenant le nom de la liste ainsi que sa date de création
* Un compteur indiquant le nombre total de thés dans la liste
* Une liste présentant chaque thé avec ses caractéristiques principales

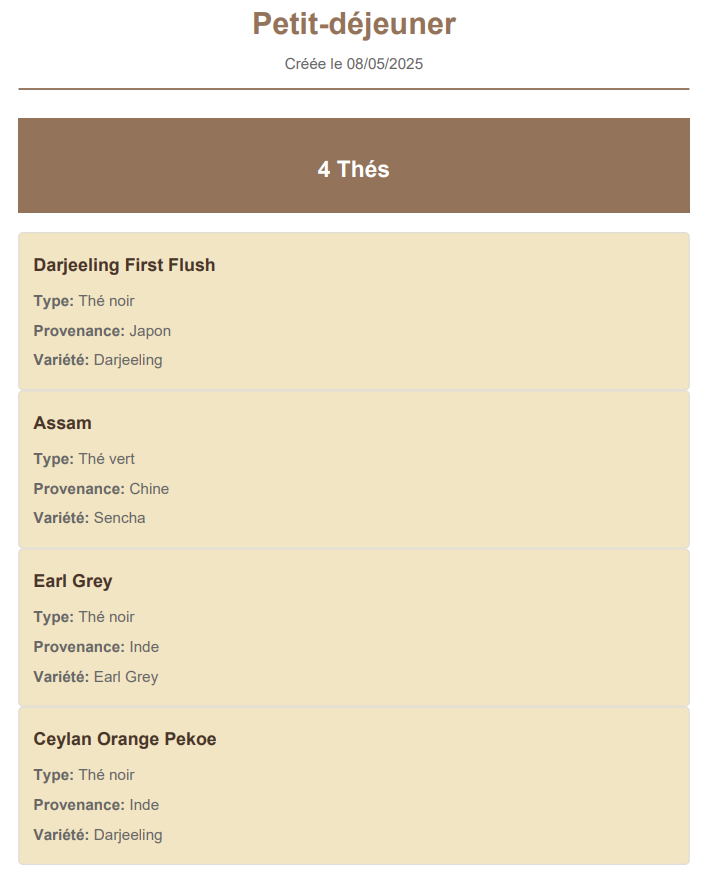
Voici un exemple de PDF généré :  


Figure 36 - visuel - PDF

Cette mise en page facilite la lecture et permet d'avoir une vue d'ensemble claire de la collection.

#### Fonctionnement

Le bouton décrit dans la partie précédente, envoie une requête à la route ‘’listes.export’’ qui est traitée par la méthode export du ‘’ListeController’’.

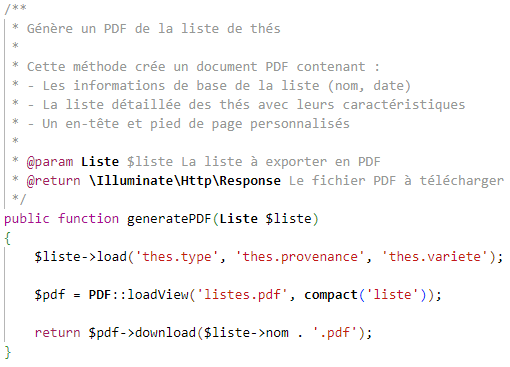
Ensuite ce ‘’ListeController’’ utilise la bibliothèque de Laravel ‘’DomPDF’’ pour transformer un template HTML en document PDF : 

Figure 37 - code - fonction - generatePDF

Cette méthode :

* Charge toutes les relations nécessaires pour afficher les détails complets des thés
* Utilise la façade PDF pour charger le template de vue ‘’listes.pdf’’ provenant du fichier ‘‘pdf.blade.php’’
* Transmet les données de la liste au template
* Retourne une réponse pour télécharger directement le fichier avec le nom de la liste

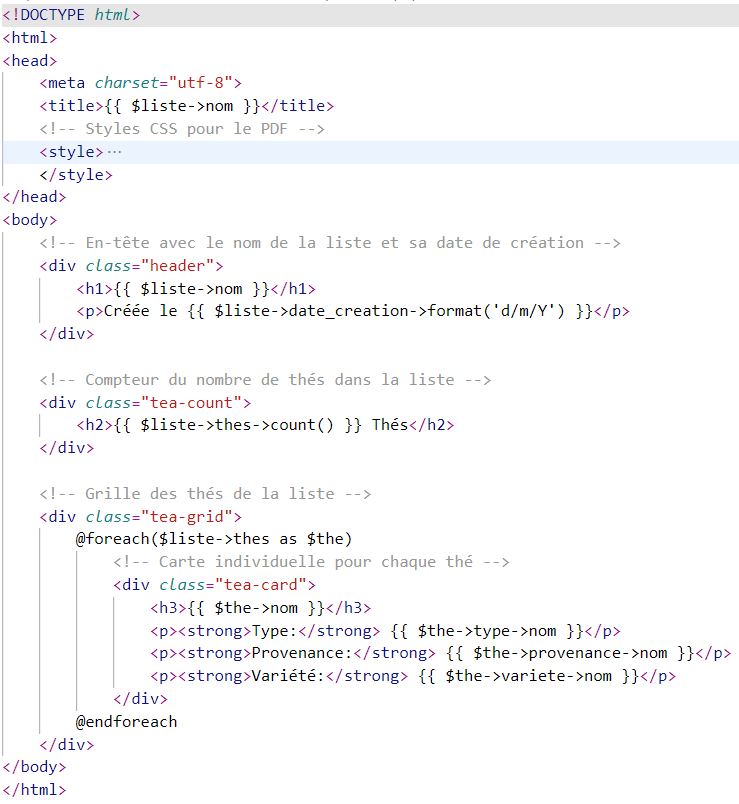
Le template ‘’listes.pdf.blade.php’’ est un document HTML avec des styles CSS intégrés, optimisé pour le rendu PDF :

Figure 38 - code - template PDF

Le template utilise une mise en page en grille CSS pour organiser les thés en liste, avec un style cohérent avec l'identité visuelle de base de l'application.

Cette méthode d'export PDF présente plusieurs avantages :

* Séparation des responsabilités : Le contrôleur gère la logique, tandis que le template gère la présentation
* Facilité de maintenance : Le template peut être modifié indépendamment du code PHP à tout moment
* Expérience utilisateur : Le téléchargement direct du fichier avec un nom approprié facilite grandement l'utilisation

En combinant un contrôleur bien structuré et un template HTML/CSS adapté, l'application offre une fonction d'export PDF élégante et professionnelle qui valorise les données souhaitées.

*Décrire la réalisation "physique" de votre projet*

* *les répertoires où le logiciel est installé*
* *la liste de tous les fichiers et une rapide description de leur contenu (des noms qui parlent !)*
* *les versions des systèmes d'exploitation et des outils logiciels*
* *la description exacte du matériel*
* *le numéro de version de votre produit !*
* *programmation et scripts: librairies externes, dictionnaire des données, reconstruction du logiciel - cible à partir des sources.*

*NOTE : Evitez d’inclure les listings des sources, à moins que vous ne désiriez en expliquer une partie vous paraissant importante. Dans ce cas n’incluez que cette partie…techniquement les problèmes rencontrés et comment les résoudre*

J’ai un dossier ‘’NeoMasson\_TPI’’qui est aussi la base de mon repos GitHub qui contient trois principaux sous-dossiers :

1 Code, qui contient le dossier principal du projet avec tous les composants Laravel et toutes les pages de codes.

2 Documentation, qui renferme tout ce qui n’est pas du code et qui pourrait servir au projet, par exemple les maquettes, des tutos d’installation ou mon rapport TPI.

3 Rendu, c’est ici que je conserve mes fichiers ZIP de rendu. Cela peut aider pour le versioning ou si un expert a besoin que je lui renvoie l’un des rendus.

### Description physique du projet

### Répertoires où le logiciel est installé :

### Liste des fichiers et description :

## Description des tests effectués

Tout au long du développement, des vérifications manuelles (‘’smoke tests’’) ont permis de s’assurer du bon fonctionnement général de l’application. Dans cette section, on se concentre sur les tests fonctionnels majeurs définis en phase de conception ainsi que sur des tests unitaires en php. Nous comparerons le résultat attendu avec le résultat réel observé.

## Tests de Fonctionnalités effectués

Voici un tableau qui reprends ces tests :

|  |  |  |  |
| --- | --- | --- | --- |
| **N°** | **Action** | **Résultat attendu** | **Résultat réel du test** |
| 1 | Tenter de s’inscrire avec un nouvel e-mail valide puis se déconnecter et se reconnecter | L’inscription crée un compte, la connexion fonctionne et l’utilisateur est redirigé vers le dashboard | Notre compte est créé et enregistré dans la base de données. Après déconnexion, il nous suffit de remettre notre adresse et notre mot de passe dans les champs de connexion cette fois ci et nous voilà reconnecté |
| 2 | Tenter d’accéder à /dashboard ou n’importe quelle page autre que celles de login sans être connecté | L’accès est refusé et il est impossible d’aller plus loin | L’accès est refusé à toutes autre URL que /login et /register et nous sommes redirigé vers la page de connexion après avoir essayé une autre page |
| 3 | Soumettre le formulaire d’ajout de thé avec un champ « prix » vide | Le formulaire est rejeté, et des messages d’erreur clairs s’affichent sous chaque champ concerné | Ce pop-up s’affiche :  Figure 39 - test - pop-up  Et le formulaire n’est donc pas validé |
| 4 | Créer un nouveau thé, puis modifier sa quantité, le consulter et enfin le supprimer | Après chaque action, les données sont correctement enregistrées / affichées / supprimées dans la base MySQL | Création, édition et suppression fonctionnent sans délai. Les changements sont instantanément visibles dans la liste et dans la table ‘’t\_the’’ via MySQL Workbench |
| 5 | Créer une liste de thés, y ajouter plusieurs thés, puis cliquer sur « Exporter en PDF » | Un fichier PDF est téléchargé, contenant la liste des thés avec leur nom | Le bouton ‘’Exporter en PDF’’ déclenche immédiatement le téléchargement d’un document structuré (titre de liste, date, tableau des thés), prêt à l’impression |

Tous les tests fonctionnels prévus ont produit le résultat attendu, confirmant la bonne mise en œuvre des fonctionnalités essentielles : authentification, sécurité d’accès, validations de formulaires, gestion CRUD et génération de PDF. Aucun écart notable n’a été relevé, ce qui valide la stabilité de la première version déployée de l’application ExceptionTea.

## Tests Unitaires effectués

Puis viennent les tests unitaires. Si l’on suit la logique Laravel, il faut faire un fichier PHP pour chaque catégorie de test unitaire.  
 Cette commande permet de créer le fichier PHP avec le bon nom :

php artisan make:test nomDuTest --unit

Ensuite, nous pouvons tous les lancer en même temps avec cette ligne :

php artisan test --filter Unit

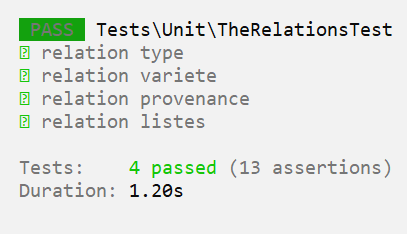
Voici le résultat d’un fichier contenant quatre tests, effectués grâce à la commande ci-dessus :  


Figure 40 - exemple – testUnit

On voit précisément quel tests passent, et en cas d’échec, un retour des détails de l’erreur est fait dans la console.

## Erreur restante

Il ne reste aucune erreur notable à la fin de ce projet.

## Liste des documents fournis

*Lister les documents fournis au client avec votre produit, en indiquant les numéros de versions*

* *le rapport de projet*
* *le manuel d'Installation (en annexe)*
* *le manuel d'Utilisation avec des exemples graphiques (en annexe)*
* *autres…*

# Conclusions

## Objectifs et Résultats

Voici un tableau qui permet de se rendre compte si des objectifs de ce projet sont atteints ou pas :

|  |  |
| --- | --- |
| *Objectif* | *Statut* |
| [*Modèle MVC*](#_Respect_du_modèle) | *✅ Atteint* |
| [*Qualité du code*](#_Qualité_et_lisibilité) | *✅ Atteint* |
| [*Authentification*](#_Processus_d’inscription_et) | *✅ Atteint* |
| [*Sécurité*](#_Sécurité_des_accès) | *✅ Atteint* |
| [*Retour utilisateur*](#_Validation_serveur_et) | *✅ Atteint* |
| [*Gestion CRUD*](#_Système_de_gestion) | *✅ Atteint* |
| [*Exportation PDF*](#_Listes_et_exportation) | *✅ Atteint* |

Comme nous pouvons le voir, pour moi, tous les objectifs ont été atteint.

*Points positifs / négatifs*

* *Difficultés particulières*

## Suites possibles pour le projet (évolutions & améliorations)

## Système de panier, commande et de paiement

Une suite logique serait d’implémenter un workflow de commande avec un panier propre à chaque utilisateur. Avant de valider leur achat, les utilisateurs pourraient ajouter plusieurs thés à ce panier, ajuster les quantités et visualiser le total. Une fois la commande passée, un processus de paiement sécurisé (via Twint ou PayPal) et l’envoi d’une facture PDF par mail complèteraient l’expérience.

## Gestion de rôles utilisateurs

Pour adapter l’accès aux différentes parties de l’application ou alors pouvoir gérer la liste des utilisateurs, il serait intéressant de définir plusieurs profils :

* Administrateur
* Simple collaborateur
* Client

Chaque rôle bénéficierait d’un niveau de permissions spécifique : seuls les administrateurs pourraient créer ou supprimer des comptes, tandis que les collaborateurs géreraient les stocks pendant que les clients pourraient visualiser les thés, en faire des listes et potentiellement en acheter (si on met en place la suite possible d’au-dessus).

## Multi-langue (i18n)

Ouvrir ExceptionTea à un public international passe par la traduction de l’interface et des descriptions de thés. L’ajout d’un sélecteur de langue permettrait de basculer facilement entre le français, l’anglais, l’allemand assurant une expérience fluide pour tous les utilisateurs.

## Import de catalogue

Pour enrichir rapidement la base, on pourrait proposer un outil d’import CSV ou Excel. Les responsables téléchargeraient simplement un fichier listant nom, type, provenance et autres détails des thés ; le système analyserait automatiquement les données, signalerait les éventuelles erreurs et intégrera les nouvelles fiches en un clic.

## Recommandations et notation

Enfin, pour personnaliser l’expérience, chaque utilisateur pourrait noter les thés et laisser un commentaire. Sur la page d’accueil, un module “Suggestions” mettrait en avant des thés similaires à ceux déjà appréciés, ou les best-sellers du moment, incitant à la découverte et renforçant l’engagement. Un système de tri par note données par les utilisateurs à un thé peut aussi être intégré à la page d’accueil.

## Bilan personnel

Lors de la réception du cahier des charges, je me suis vu un peu sceptique à l’idée de travailler sur des thés, ce sujet ne m’inspirant pas particulièrement. Pourtant, je me suis rapidement rendu compte que cela n’avait finalement que peu d’impact sur le plan technique. Le site web que j’ai développé peut tout aussi bien pu être adapté à n’importe quel type de produit ou d’objet.

Ce projet m’a permis d’approfondir mes connaissances, même dans un domaine que l’on aborde déjà en cours, comme le développement web. J’ai notamment découvert Laravel, un framework que je ne connaissais pas avant et que j’ai trouvé particulièrement intéressant et bien structuré. Je compte d’ailleurs le réutiliser dans de futurs projets.

Même si, à la base, la programmation n’est pas mon domaine de prédilection, je suis tout de même fier du résultat de ce TPI. Cela m’a permis de sortir de ma zone de confort et de voir de quoi je suis capable.

# Annexes

## Aides externes

Durant ce projet, j’ai utilisé différents outils qui m’ont permis de simplifier certaines tâches :

* Reverso.net et ChatGPT 4.0 m’ont permis de corriger mon orthographe et d’améliorer ma formulation en soulignant mes fautes ou en me donnant des conseils de rédaction.
* Looping.exe m’a facilité la création d’un script SQL à partir de mon MCD.
* Claude 3.5 Sonnet pour quand je n’arrivais pas à résoudre une erreur dans mon code, que je ne comprenais pas un snippet provenant d’internet ou pour implémenter une fonction complexe. Je lui décrivais l’erreur ou la fonction souhaitée puis il m’expliquait de manière claire avec des exemples comment je pouvais résoudre le problème ou implémenter la fonction.
* Le logo du site web a été générer à partir de l’API OpenAI – Sora. Ce fut très intéressant de voir comment la génération d’image à évoluer et à quel point nous pouvons lui demander des corrections précises.

## Résumé du rapport du TPI

## Situation de départ

ExceptionTea SA, spécialisée dans les thés d’exception à Lausanne, souhaitait moderniser leur manière d’inventoriser et d’administrer leurs thés, et cela, sans utiliser d’outils génériques existants, jugé inadaptés à la gestion de ses produits rares. L’entreprise demande donc une solution sur-mesure, moderne et sécurisée, capable de remplacer ces outils et d’optimiser le suivi des stocks.

## Mise en œuvre

Le projet a consisté à développer une application web en PHP, basée sur le framework Laravel et stylée avec Tailwind CSS, selon une architecture MVC stricte. Le kit Laravel Breeze a fourni instantanément le squelette des pages et contrôleurs d’authentification (inscription, connexion), tandis que Dompdf sert à l’export PDF. La base MySQL, conçue à partir d’un MCD et traduite en MLD puis déployée via des migrations Eloquent, assure la cohérence des entités (thés, types, variétés, provenances). Le développement a suivi la méthode des six pas, répartie en verbes, s’informer, planifier, décider, réaliser, contrôler et évaluer.

## Résultats

L’application offre désormais un accès authentifié à un tableau interactif listant tous les thés, avec recherche instantanée, filtres (type, provenance), tri par colonne et opérations CRUD complètes. Les utilisateurs peuvent créer des listes de thés personnalisées et les exporter en PDF. Plusieurs pistes d’évolution ont par ailleurs été identifiées : intégration d’un panier et d’un module de paiement, gestion fine des rôles, internationalisation de l’interface, import massif de catalogue et module de recommandations.

## Sources – Bibliographie

* 1. <https://github.com>
  2. <https://stackoverflow.com>
  3. <https://www.reverso.net/orthographe/correcteur-francais/>
  4. <https://dev.mysql.com/downloads/installer/>
  5. <https://laravel.com/docs/12.x>
  6. <https://tailwindcss.com/>
  7. <https://tailwindcss.com/docs/installation/framework-guides/laravel/vite>
  8. <https://nodejs.org/en/download>
  9. <https://getcomposer.org/>
  10. <https://www.reddit.com/r/laravel/comments/155cw2e/does_anyone_here_prefer_phpunit_to_pest/?rdt=35242>
  11. <https://tecfa.unige.ch/guides/tie/html/mysql-intro/mysql-intro-7.html>
  12. <https://louisvandevelde.be/index.php?dos=my&fic=meris>
  13. <https://www.figma.com/>
  14. <https://manjuparkavi.medium.com/10-essential-javascript-snippets-every-developer-should-know-6c68cabaa082>
  15. <https://www.solidpepper.com/blog/modele-physique-de-donnees-mpd-definition-enjeux-exemple>
  16. <https://redketchup.io/color-picker>
  17. <https://kinsta.com/fr/blog/laravel-eloquent/>
  18. <https://johackim.com/tailwind-css>
  19. <https://www.pairform.fr/comprendre-lutilisation-don-delete-cascade-en-sql.html>
  20. <https://learn.microsoft.com/fr-fr/sql/t-sql/statements/alter-schema-transact-sql?view=sql-server-ver16>
  21. <https://blog-gestion-de-projet.com/strategie-de-test/>
  22. <https://openclassrooms.com/forum/sujet/enregistrer-une-liste-dans-une-bdd>
  23. <https://fr.vecteezy.com/png-gratuit/>
  24. <https://www.afci-ju.ch/fichiers/Planification-par-la-mthode-des-6-tapes-15.pdf>
  25. <https://heroicons.com/>
  26. <https://sora.chatgpt.com>
  27. <https://www.v-labs.fr/glossaire/orm-object-relational-mapping/>
  28. [https://www.puce-et-media.com/utiliser-pdo-pour-gerer-plusieurs-types-de-bases-de-donnees-en-php/](https://www.puce-et-media.com/utiliser-pdo-pour-gerer-plusieurs-types-de-bases-de-donnees-en-php/#:~:text=PDO%20est%20une%20extension%20PHP,en%20utilisant%20une%20API%20unique)
  29. <https://www.ibm.com/fr-fr/topics/>
  30. <https://datascientist.fr/blog/le-guide-ultime-du-crud-pour-les-developpeurs-web/>
  31. <https://pieces.app/collections/javascript>
  32. <https://dev.to/abhirajb/series/15463>
  33. <https://www.managedserver.eu/what-are-character-sets-and-collations-in-mysql-and-mariadb/>

## Glossaire

* **API** (Interface de Programmation d’Applications) : Interface définissant les méthodes d’échange entre deux systèmes ou services différents, facilitant l’intégration de fonctionnalités.
* **Blade** : Moteur de templates de Laravel. Les fichiers .blade.php mêlent HTML et directives (@if, @foreach, etc.) pour générer le rendu côté serveur.
* **Breeze** : Kit officiel Laravel fournissant l’authentification (inscription, connexion, réinitialisation) avec des vues Blade et Tailwind CSS prêtes à l’emploi.
* **Clé primaire composite** : Clé primaire constituée de plusieurs colonnes dans une table relationnelle, garantissant qu’aucune combinaison de ces champs ne se répète (ex. (fk\_id\_liste, fk\_id\_the)).
* **Composer** : Gestionnaire de dépendances PHP, utilisé pour installer et mettre à jour les bibliothèques requises par un projet Laravel (via composer.json).
* **CRUD** : Acronyme de **Create, Read, Update, Delete** ; désigne les quatre opérations de base sur une ressource (par exemple : créer, consulter, modifier et supprimer un enregistrement).
* **Diagramme de Gantt** : Représentation graphique d’un planning projet sous forme de barres horizontales, indiquant la durée et l’enchaînement des tâches.
* **Dompdf** : Librairie PHP qui convertit du HTML/CSS en PDF, idéale pour générer des exports imprimables (listes de thés, factures…).
* **GitHub Desktop** : Application de bureau simplifiant l’usage de Git et de GitHub, pour cloner, committer et synchroniser un repos sans passer par la ligne de commande.
* **IDE** (Environnement de Développement Intégré) : Logiciel tout-en-un pour coder, tester et déboguer (ex. : Visual Studio Code, PhpStorm).
* **Laravel** : Framework PHP suivant le modèle MVC (Modèle–Vue–Contrôleur), offrant gestion des routes, ORM Eloquent, authentification, migrations et plus.
* **Looping.exe** : Outil de modélisation Merise (MCD/MLD) gratuit, utilisé pour concevoir graphiquement la structure d’une base de données relationnelle.
* **Middleware** : Composant intermédiaire qui s’exécute entre la requête HTTP et le contrôleur pour appliquer des vérifications ou transformations (authentification, filtrage…).
* **MySQL** : Système de gestion de base de données relationnelle (SGBDR) très répandu, basé sur le langage SQL pour stocker et interroger les données.
* **Workbench** : Interface graphique officielle de MySQL pour administrer les bases, exécuter des requêtes SQL et modéliser des schémas.
* **NodeJS** : Environnement d’exécution JavaScript côté serveur, permettant d’utiliser JavaScript au-delà du navigateur.
* **NPM** (Node Package Manager) : Gestionnaire de paquets pour Node.js, utilisé pour installer et gérer les bibliothèques JavaScript (ex. : Vite, Tailwind CSS).
* **PDO** (PHP Data Objects) : Extension PHP fournissant une interface unifiée pour accéder à différentes bases de données via PDO, avec support des requêtes préparées.
* **Pest** : Framework de tests PHP simple et élégant, alternatif à PHPUnit, utilisé pour écrire et exécuter rapidement des tests unitaires et fonctionnels.
* **PHP** : Langage de script côté serveur, principal pour le développement web avec Laravel, exécuté par un interpréteur tel que PHP-CLI ou intégré à un serveur HTTP.
* **Reverso.net** : Service en ligne de correction de texte aussi pour améliorer la qualité des textes multilingues.
* **SGBD** (Système de Gestion de Base de Données) : Logiciel chargé de stocker, organiser et sécuriser les données, et de traiter les requêtes SQL.
* **Snippets éprouvés** : Extraits de code testés et fiables, que l’on peut copier-coller pour gagner du temps et éviter les erreurs sur des fonctionnalités courantes.
* **Tailwind CSS** : Framework utilitaire CSS, offrant des classes prédéfinies (marges, couleurs, typographie…) à appliquer directement dans le HTML.
* **TPI** : Projet individel, examen pratique de fin de CFC d’informaticien (~90 h), comprenant réalisation technique et documentation.
* **Vite** : Outil de bundling et de développement web moderne, associé à Laravel via laravel-vite-plugin, pour compiler JavaScript et CSS rapidement.
* **Visual Studio Code** : Éditeur de code gratuit et extensible de Microsoft, très populaire pour le développement web.
* **Wireframe** : Maquette basse fidélité sans styles ni couleurs, servant à définir la structure et l’agencement des éléments d’une page avant le design final.

## Table des illustrations

[Figure 1 - exemple - Séquence de planification 5](#_Toc198736327)

[Figure 2 exemple - Diagramme Gantt 5](#_Toc198736328)

[Figure 3 - graphique des tâches planifiées 6](#_Toc198736329)

[Figure 4 - maquette - S'enregistrer 9](#_Toc198736330)

[Figure 5 - maquette - S’authentifier 9](#_Toc198736331)

[Figure 6 - maquette - Accueil 10](#_Toc198736332)

[Figure 7 - maquette - Détails du Thé 10](#_Toc198736333)

[Figure 8 - maquette - Ajouter un thé 11](#_Toc198736334)

[Figure 9 - maquette – Listes 12](#_Toc198736335)

[Figure 10 - maquette - Détails d'une liste 12](#_Toc198736336)

[Figure 11 - BDD – MCD 13](#_Toc198736337)

[Figure 12 - BDD – MLD 13](#_Toc198736338)

[Figure 13 - BDD – MPD 14](#_Toc198736339)

[Figure 14 - solution - navbar – code 18](#_Toc198736340)

[Figure 15 - barre de recherche 19](#_Toc198736341)

[Figure 16 - affichage - recherche 19](#_Toc198736342)

[Figure 17 - visuel - filtrage 20](#_Toc198736343)

[Figure 18 - exemple - filtrage 20](#_Toc198736344)

[Figure 19 - exemple - tri 21](#_Toc198736345)

[Figure 20 - code - fonction – sortTable 22](#_Toc198736346)

[Figure 21 - code – section filtrage de recherche - HTML 23](#_Toc198736347)

[Figure 22 - code - début du script 23](#_Toc198736348)

[Figure 23 - code - fonction - filterTable 24](#_Toc198736349)

[Figure 24 - code - EventListener 24](#_Toc198736350)

[Figure 25 - visuel - listes déroulantes 25](#_Toc198736351)

[Figure 26 - visuel - listes déroulantes ouvertes 25](#_Toc198736352)

[Figure 27 - visuel - pop-up ajout 26](#_Toc198736353)

[Figure 28 - code - bouton - listes déroulantes 26](#_Toc198736354)

[Figure 29 - code - liste 27](#_Toc198736355)

[Figure 30 -code - fonction - toggleList 27](#_Toc198736356)

[Figure 31 - code - fonction - Pop-up 28](#_Toc198736357)

[Figure 32 - code - fonction - updateListe 29](#_Toc198736358)

[Figure 33 - visuel - Exporter PDF 29](#_Toc198736359)

[Figure 34 - visuel - PDF 30](#_Toc198736360)

[Figure 35 - code - fonction - generatePDF 31](#_Toc198736361)

[Figure 36 - code - template PDF 32](#_Toc198736362)

[Figure 37 - test - pop-up 34](#_Toc198736363)

[Figure 38 - manuel - php –version 41](#_Toc198736364)

[Figure 39 - manuel - composer --version 41](#_Toc198736365)

[Figure 40 - manuel - versions node & npm 41](#_Toc198736366)

[Figure 41 - manuel – dossier 42](#_Toc198736367)

[Figure 42 - manuel - Mysql -- version 42](#_Toc198736368)

[Figure 43 - manuel - ShowDatabases 44](#_Toc198736369)

[Figure 44 - manuel - composer run dev 46](#_Toc198736370)

*Liste des livres utilisés (Titre, auteur, date), des sites Internet (URL) consultés, des articles (Revue, date, titre, auteur) … Et de toutes les aides externes (noms)*

## Mise en place de l’environnement de projet

Dans ce manuel nous allons suivre le processus d’installation de tout l’environnement de projet, avec Laravel, ses dépendances et la base de données.

## Installer PHP

Dans un premier temps nous allons installer PHP, dans notre cas pour Windows, depuis l’adresse : <https://windows.php.net/download/>

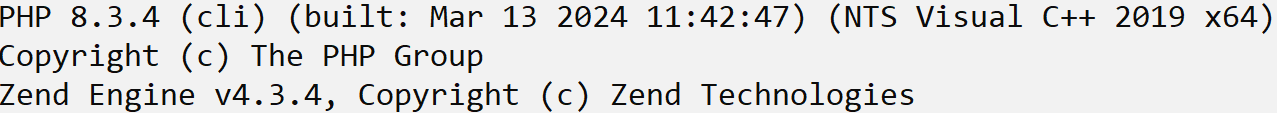
Si vous rencontrez une incompatibilité, mettez à jour les Visual C++ depuis : <https://learn.microsoft.com/visualstudio/releases/2022/support/vc-redist>  
Vous pouvez vérifier l’installation depuis le terminal avec la commande : php –version  
vous devriez avoir un résultat similaire :  


Figure 41 - manuel - php –version

Il faut ensuite configurer le fichier php.ini pour activer les extensions nécessaires au bon fonctionnement de Laravel.

Décommentez ces lignes :

extension=fileinfo

extension=pdo\_mysql

Ces extensions permettent la gestion des types de fichiers et la connexion à une base de données MySQL via PDO.

## Installer composer

Rendez-vous sur le site officiel : <https://getcomposer.org/>

Téléchargez l’installateur Windows puis lancez-le.

Vérifiez l’installation avec :

composer –version

Résultat attendu de la commande :

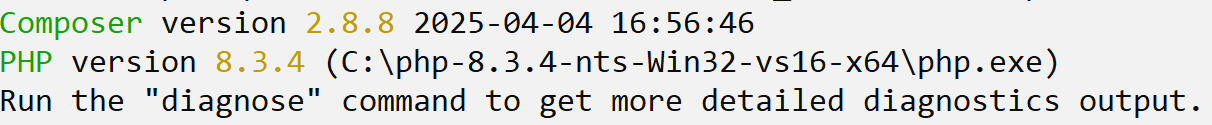


Figure 42 - manuel - composer --version

## Installer Node.js et NPM

Téléchargez Node.js (incluant NPM) depuis ce lien : <https://nodejs.org/fr>

Installez et vérifiez les installations :

node --version

npm –version

Le terminal devrait vous retournez quelque chose comme :

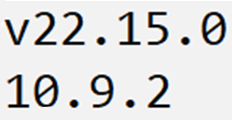


Figure 43 - manuel - versions node & npm

## Créer le projet Laravel

Pour initialiser notre dossier de projet Laravel, déplacez-vous dans le dossier où vous souhaitez stocker votre code et exécutez :

composer create-project laravel/laravel ExceptionTea

Si vous souhaitez un autre nom de projet remplacez ‘’ExceptionTea’’ par le vôtre.

La commande va ensuite installer une longue liste de dépendances, il suffit de patienter un peu.

Puis pour accéder à votre dossier :

…cd ExceptionTea

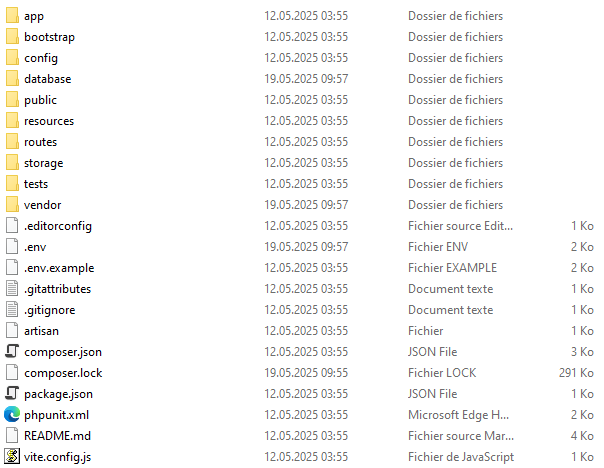
Vous verrez que le dossier a déjà une arborescence de fichier structurée et prête à l’emploi :  


Figure 44 - manuel – dossier

## Installer MySQL

Téléchargez l’installateur Windows (MySQL Community Server) : <https://dev.mysql.com/downloads/installer/>

Lancez-le, sélectionnez MySQL Server et configurez un mot de passe ‘’root’’ (si demandé).

Vérifiez l’installation dans un terminal :

mysql --version

Vous devriez avoir une ligne similaire :  


Figure 45 - manuel - Mysql -- version

## Installer Breeze

Depuis le dossier du projet, installez le package de Breeze:

composer require laravel/breeze –dev

php artisan breeze:install blade

Compilez les actifs pour appliquer les styles :

npm install

npm run dev

## Configurer l’environnement Laravel et se connecter à la DB

Dupliquez le fichier ‘’.env.example’’ :

copy .env.example .env

Générez la clé de chiffrement :

php artisan key:generate

Éditez le fichier ‘’.env’’, dans la partie base de données de celui-ci et remplir les informations suivantes :

DB\_CONNECTION=mysql

DB\_HOST=127.0.0.1

DB\_PORT=3306

DB\_DATABASE=exceptiontea

DB\_USERNAME=root

DB\_PASSWORD=XXX

Adaptez bien sur ces différentes données à votre environnement.  
Choisissez un mot de passe ou réutiliser le même que lors de [l’installation de MySQL](#_Installer_MySQL), celui pour votre utilisateur ‘’root’’.

## Créer la base de données MySQL

Allez dans le dossier de votre instance MySQL et connectez-vous grâce à :

mysql -u root -p

Entrez votre mot de passe root.

Puis pour créer la DB exécutez :

CREATE DATABASE exceptiontea

CHARACTER SET utf8mb4

COLLATE utf8mb4\_unicode\_ci;

Les deux dernières lignes spécifient que la DB utilisera le jeu de caractères ‘’utf8mb4’’ (pour prendre en charge les caractères spéciaux, y compris les émojis) et le classement ‘’utf8mb4\_unicode\_ci’’ (pour des textes insensibles à la casse).

Vérifiez qu’elle existe :

SHOW DATABASES;

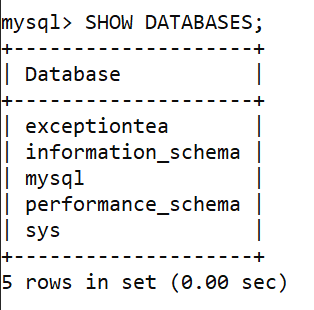
****

Figure 46 - manuel - ShowDatabases

Votre nouvelle base de données devrait être présente dans la liste ci-dessus.

## Installer Dompdf

Installez la librairie :

composer require barryvdh/laravel-dompdf

Publiez la configuration :

php artisan vendor:publish --provider="Barryvdh\DomPDF\ServiceProvider"

Vous pouvez ensuite utiliser DomPDF dans votre code.

## Migrations pour la base de données

Au lieu de faire directement du SQL pour créer les tables, attributs et relations, Laravel utilise un ORM nommé Eloquent et un système de migrations qui permettent de versionner et de partager la structure de la base de données comme du code PHP.

Chaque migration décrit la création ou la modification d’une table ; en lançant :

php artisan migrate

Laravel traduit ces classes en commandes SQL, ce qui permet d’avoir une base de données relationnelles même si nous savons peu des requêtes SQL.

#### Génération des modèles

Pour chaque entité de la base, on crée un **Model** Eloquent et une **migration** depuis le terminal Windows ou PowerShell.

Par exemple dans ce projet, pour créer nos modèles nous avons :

php artisan make:model Provenance -m

php artisan make:model Type -m

php artisan make:model Variete -m

php artisan make:model The -m

php artisan make:model Liste -m

php artisan make:migration create\_contient\_table

Grace au flag ‘’-m’’ ces commandes vont créer des fichiers de migrations dans ‘’$votre\_dossier\_de\_projet’’/database/migrations. Puis il faudra aller éditer ces fichiers pour y renseigner les différents attributs, relations et type de données de chaque table.

Voici un exemple de fichiers de migration pour la table t\_provenance :

public function up()

{

Schema::create('t\_provenance', function (Blueprint $table) {

$table->id('id\_provenance');

$table->string('nom');

$table->timestamps();

});

}

Puis pour effectuer ces migrations jusqu’à la base de données on utilise la commande :

php artisan migrate

## Lancement de l’application

Nous voilà aux dernières étapes de la configuration de notre environnement.

Afin d’être sûr de ne manquer d’aucun composant nous allons installer toutes les librairies listées dans ‘’composer.json’’ nécessaire au bon fonctionnement de notre projet ainsi que les dépendances JavaScript, et ce, avec la commande :

composer install

npm install

Si vous souhaitez compiler une modification récente utilisez :

npm run dev

Puis pour finalement lancer votre serveur de développement et compiler TailWindCSS & Vite utilisez la commande :

composer run dev

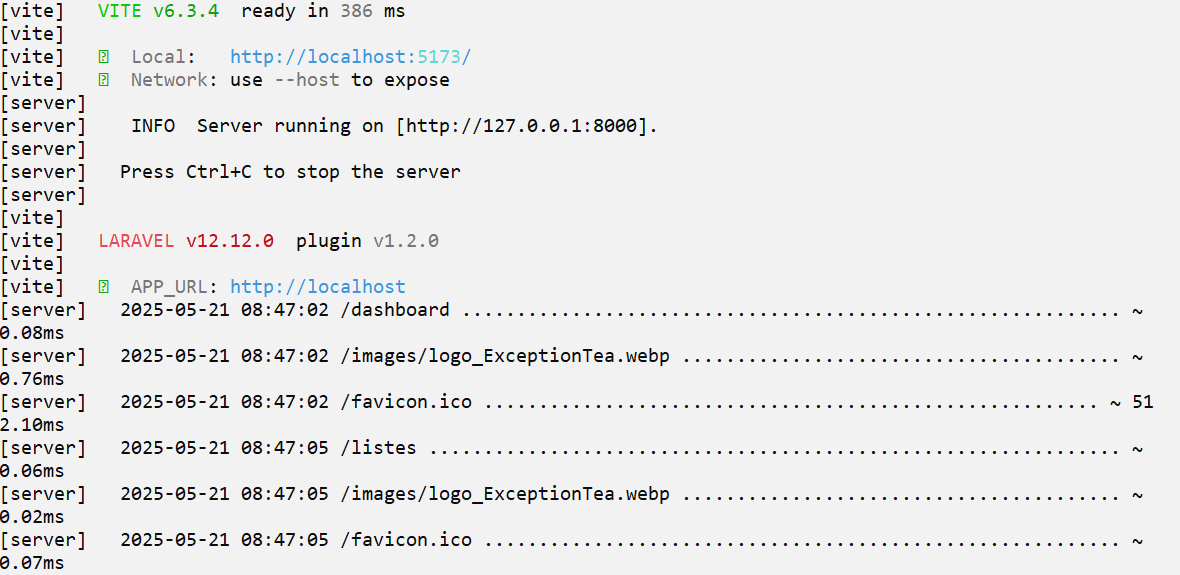
Votre instance locale se lancer alors et affiche une sortie similaire à ceci :   


Figure 47 - manuel - composer run dev

Les premières lignes nous donnent l’URL d’accès à notre site avec son port par défaut :‘’http://localhost :5173’’.  
C’est avec cette adresse entrée dans votre navigateur que vous pourrez visualiser votre application web.  
Quelques lignes plus bas, on aperçoit la version de laravel qui est en cours d’exécution mais surtout les logs en temps réel générés par le serveur, ce qui est très utile pour déboguer.  
  
**Bravo !**

Votre environnement est désormais prêt. Ouvrez le dossier de projet dans votre éditeur (par exemple Visual Studio Code) : à chaque ‘’Ctrl + S’’, Vite recompile automatiquement et actualise le navigateur pour voir vos modifications en direct. Vous êtes fin prêt à démarrer le développement de votre application !

## Journaux de travail

## Prérequis ?