

LIVRABLE NON TECHNIQUE PROJET DEVELOPPEMENT WEB ET OUTILS GENIE LOGICIEL

GROUPE 11



MEMBRES DU GROUPE

TCHATCHOUA GAMGA Ariane Eva (chef de projet)

TESSA TEKEU Ralph Axel

GUENDJIO NGASSA Yvan

SCHULE SADO Franck Steven

NKE GOUETH Emmanuel Yvan

TCHISSAMBOU JOSSY Steven

24|10|2024

TABLES DE MATIERES

***TABLE DES MATIÈRES***

I/ Description de l'architecture du projet

* + 1- Choix des technologies
  + 2- Modules et services

II/ Guide d'installation et d'utilisation

III/ Schémas de la base de données

IV/ Charte graphique de l'application

* **I/ Description de l'architecture du projet**
* **1.Choix des technologies**
* **1.1. Backend**
* **Langage** : PHP, JavaScript
* **Framework**: Laravel
* **Bibliothèques** : Boostrap.js & Boostrap.css
* **Base de données** : MySQL
* **Motif de choix** :
  + **Laravel** est un framework PHP moderne qui se distingue par sa syntaxe élégante et expressive, facilitant le développement d'applications web. Il offre une multitude d'outils intégrés, tels que l'Eloquent ORM, Artisan CLI et le moteur de templates Blade, qui simplifient les tâches courantes et améliorent la productivité des développeurs. Sa sécurité robuste, son architecture MVC, et sa capacité à évoluer avec les besoins des projets en font un choix privilégié pour des applications allant de petite à grande envergure.
  + **MySQL** est un système de gestion de base de données relationnelle (SGBDR) open source, largement utilisé pour le stockage et la gestion de données permettant de créer, lire, mettre à jour et supprimer des données à l'aide du langage SQL. Grâce à sa compatibilité avec divers systèmes d'exploitation et son intégration facile avec des langages de programmation comme PHP et Python, MySQL est souvent associé à des frameworks comme Laravel et des plateformes comme WordPress.
* **1.2. Frontend**
* **Langage** : JavaScript, Html & CSS
* **Framework** : Boostrap
* **Motif de choix** :
  + **Bootstrap** est un framework CSS populaire qui facilite la création de sites web réactifs et esthétiques. Avec sa vaste bibliothèque de composants préconstruits et sa documentation exhaustive, il permet aux développeurs de concevoir rapidement des interfaces utilisateur cohérentes et attrayantes. Sa personnalisation aisée, sa compatibilité avec divers navigateurs, et son intégration fluide avec d'autres technologies en font un outil incontournable pour le développement front-end.
* **1.3. Base de données**
* **Système de gestion de base de données (SGBD)** : MySQL
* **Motif de choix** : Les fonctionnalités relationnelles et la gestion transactionnelle de **MySQL** sont essentielles pour assurer l'intégrité des données lors des opérations de fidélité et de parrainage.
* **2. Modules et services**
* **2.1. Module de Fidélité**
* **Objectif : Gérer le système de points de fidélité pour les utilisateurs.**
* **Fonctionnalités :**
* Gestion des Points :
* Attribution de points à chaque commande.
* Déduction de points lors de l'utilisation pour le paiement.
* Historique des Transactions :
* Affichage d'un tableau récapitulatif des points gagnés et utilisés, avec des détails sur chaque commande (date, montant, points attribués).
* **2.2. Module de Parrainage**
* **Objectif : Encourager les utilisateurs à parrainer de nouveaux clients.**

**Fonctionnalités :**

* Génération de Codes de Parrainage :
* Création d'un code unique pour chaque utilisateur, qui peut être partagé avec des filleuls.
* Suivi des Filleuls :
* Affichage de la liste des utilisateurs parrainés et leur statut (inscrit, première commande effectuée).
* Indication si une récompense a été attribuée pour chaque filleul.
* **2.3. Module Admin**
* **Objectif : Permettre à l'administrateur de gérer le programme de fidélité et de parrainage.**
* **Fonctionnalités :**
* Statistiques :
* Affichage de statistiques globales sur les points de fidélité et le parrainage (nombre d'utilisateurs, points échangés, etc.).
* Gestion des Paramètres :
* Ajustement des paramètres du programme, tels que le taux de conversion des points et la durée de validité des points.
* Validation des Réclamations :
* Visualisation des demandes de réclamation de points par les utilisateurs et possibilité de les valider ou de les rejeter.
* **2.4. Services**
* **Service d'Authentification**
* **Objectif : Gérer l'identité et les sessions des utilisateurs.**
* **Fonctionnalités :**
* Inscription et Connexion :
* Création de comptes utilisateurs avec validation des données.
* Authentification des utilisateurs avec gestion des sessions ou des tokens JWT.
* Gestion des Profils :
* Permettre aux utilisateurs de modifier leurs informations personnelles.
* Récupération de Mot de Passe :
* Fonctionnalité pour réinitialiser le mot de passe en cas d'oubli.
* **2.5 Structure des API**

Les APIs sont basées sur l'architecture REST pour faciliter les interactions avec le backend :

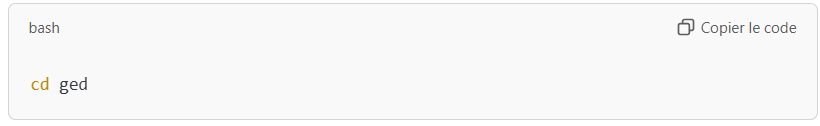
* **Endpoints pour récupérer les informations de l'utilisateur connecté en envoyant une requête** :
* function (Request $request) {return $request->user(); });
* **II/ Guide d'installation et d'utilisation**
* **1 Installation en local**
* **Prérequis :**
* **Laravel** installé (version 9.52.16)
* **MySQL** installé et configuré
* **Étapes d’installation :**

1. **Cloner le projet** :



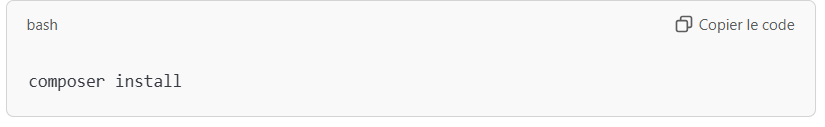
1. **Naviguer dans le dossier du projet**

Accédez au répertoire du projet cloné :

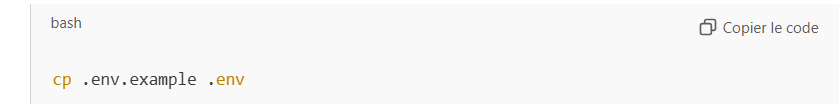
****

1. **Installer les dépendances**

Utilisez Composer pour installer les dépendances nécessaires :



1. **Configurer l'environnement**

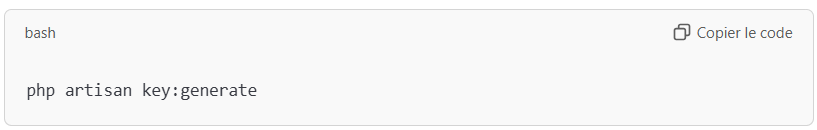


Copiez le fichier d'environnement :

Ouvrez le fichier.env. et configurez les paramètres de votre base de données (DB\_HOST, DB\_DATABASE, DB\_USERNAME, DB\_PASSWORD).

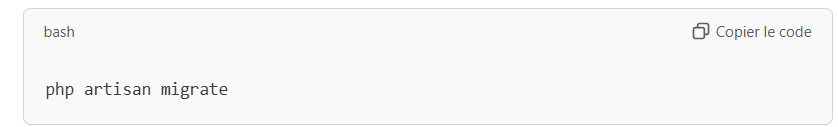
1. **Générer la clé d'application**

Exécutez la commande suivante pour générer une clé unique pour votre application :

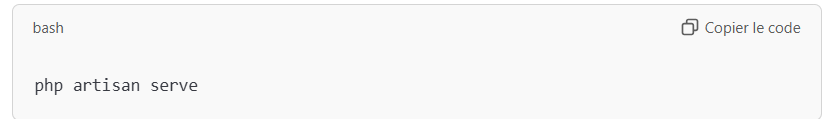
****

1. **Exécuter les migrations**

Créez les tables de la base de données en exécutant :

****

1. **Démarrez le serveur local :**

****

* **III/ . Schémas de la base de données**
* **3.1 Tables principales**
* **3.1.1 Table : Utilisateurs (users)**

| **Champ** | **Type** | **Description** |
| --- | --- | --- |
| user\_id | INT | Clé primaire, ID unique de l'utilisateur |
| username | VARCHAR(50) | Nom d'utilisateur |
| email | VARCHAR(100) | Adresse email |
| phone\_number | VARCHAR(20) | Numéro de téléphone |
| fidélité\_points | INT | Solde des points de fidélité |
| inscription\_date | INT | ID du parrain (nullable) |
| account\_status | DATETIME | Date d'inscription |
| parrain\_id | ENUM | Statut du compte (actif/inactif) |
| role | INT | Rôle |
| card\_number | VARCHAR(20) | Numéro de carte |
| user\_password | INT | Mot de passe |
| user\_age | INT | Age |

* **3.1.2 Table : Commandes**

| **Champ** | **Type** | **Description** |
| --- | --- | --- |
| order\_id | INT | Clé primaire, ID unique de la commande |
| user\_id | INT | Référence vers l'ID utilisateur |
| montant\_total | DECIMAL | Montant total de la commande |
| nombre\_points\_accumules | INT | Points gagnés sur cette commande |
| order\_date | DATETIME | Date de la commande |

* **3.1.3 Table : Referrals**

| **Champ** | **Type** | **Description** |
| --- | --- | --- |
| parrain\_id | INT | Clé primaire |
| filleul\_id | INT | ID du filleul |
| parrainage\_date | INT | Date de parrainage |
| etat\_de\_récomponse | ENUM | Etat de récompense |

**3.1.4 Table Administrateur**

| **Champ** | **Type** | **Description** |
| --- | --- | --- |
| administrateur\_id | INT | Clé primaire |
| email\_admin | INT | Email de l’administrateur |
| password\_admin | INT | Mot de passe de l’administrateur |
| nom\_admin | ENUM | Nom |

* **3.1.5 Table : Affiche**

| **Champ** | **Type** | **Description** |
| --- | --- | --- |
| affiche\_id | INT | Clé primaire |
| type\_affiche | INT | Type d’affiche |
| description | INT | Description |

* **3.1.6 Table : Boissons**

| **Champ** | **Type** | **Description** |
| --- | --- | --- |
| boisson\_id | INT | Clé primaire |
| nom\_boisson | INT | Nom de la boisson |
| Prix\_boisson | INT | Prix de la boisson |
| Quantité | INT | Quantité |

* **3.1.7 Table : Historique de commande**

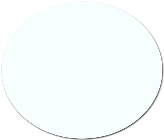
| **Champ** | **Type** | **Description** |
| --- | --- | --- |
| id\_historique | INT | Clé primaire |
| order\_id | INT | ID du filleul |
| plat\_id | INT | Date de parrainage |
| nom\_plat | ENUM | Etat de récompense |
| prix\_unitaire | DECIMAL | Prix |
| quantite | INT | Quantité |
| total | ENUM | Total |
| date\_commande | DATETIME | Date de commande |

* **3.1.8 Table : Plats**

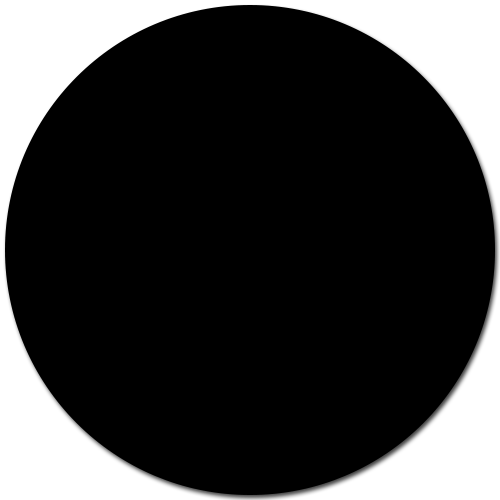
| **Champ** | **Type** | **Description** |
| --- | --- | --- |
| plat\_id | INT | Clé primaire, ID unique de la commande |
| nom\_plat | INT | Référence vers l'ID utilisateur |
| prix | DECIMAL | Montant total de la commande |
| quantite | INT | Points gagnés sur cette commande |
| statut\_de\_livraison | DATETIME | Date de la commande |
| description\_plat | INT | Description |

**3.1.9 Table Réclamations**

| **Champ** | **Type** | **Description** |
| --- | --- | --- |
| reclamation\_id | INT | Clé primaire |
| user\_id | VARCHAR (20) | ID de l’utilisateur |
| contenu\_reclamation | INT | Contenu de la réclamation |

* **3.2 Relations**
* HISTORIQUE\_COMMANDE a une relation de type 0,N avec PLAT, où PLAT est lié à HISTORIQUE\_COMMANDE par l'attribut plat\_id.
* PLAT a une relation de type 1,N avec COMMANDER\_PLAT, où COMMANDER\_PLAT est lié à PLAT par l'attribut plat\_id.
* RECLAMATION a une relation de type 1,N avec AFFICHE, où AFFICHE est lié à RECLAMATION par l'attribut affiche\_id.
* COMMANDE a une relation de type 1,N avec ENREGISTRER, où ENREGISTRER est lié à COMMANDE par l'attribut order\_id.
* ENREGISTRER a une relation de type 1,N avec DETAILLER, où DETAILLER est lié à ENREGISTRER.
* INCLUDE a une relation de type 1,N avec UTILISATEUR, où UTILISATEUR est lié à INCLUDE par l'attribut user\_id.
* PARRAINAGE a une relation de type 0,N avec PARRANER, où PARRANER est lié à PARRAINAGE par l'attribut parrain\_id.
* PARRAINAGE a une relation de type 0,N avec PASSER, où PASSER est lié à PARRAINAGE par l'attribut filleu\_id.
* ADMINISTRATEUR a une relation de type 1,N avec COMMANDER, où COMMANDER est lié à ADMINISTRATEUR par l'attribut nom\_admin..
* **IV/ Charte graphique de l'application**
* **4.1 Police**
* **Police principale** : *Arial* (Sans-serif)
* **Police secondaire** : *Dancing Script*
* **4.2 Palette de couleurs**

| **Couleur** | **Code Hexadecimal** |
| --- | --- |
|  | #CFBD97 |
| Beige | #3A3A3A |
| Noir | #0000 |
| Blanc | #FFFF |



### 