CAHIER DE CHARGE PROJET : Mon Miam Miam

Propriétaire : **ZeDuc@Space** - x - **DNx**

Date : 16/09/2024

Version : V1.0.2

1. Introduction 

ZeDuc@Space est le restaurant préféré des étudiants de l’institut UCAC-ICAM de yansoki. Situé au sommet de la résidence la terrasse, c’est devenu l’endroit par excellence des étudiants car en dehors du bon miam miam qui est proposé, il y’a également une ambiance de divertissement déjà en profitant du vent et de la vue sur la Dibamba mais aussi la disposition d’une télévision avec Canal pour les matchs de champions League et d’un billard pour les amorceurs du sport de luxe.

1. Contexte et objectif du projet
2. Contexte

Victime de son succès, les employés ont du mal à satisfaire efficacement le flux des commandes qui sont jusqu’ici effectué par messagerie et appel. D’autre part, Mr miam miam le propriétaire souhaite offrir à ses clients une expérience magique avec la possibilité d’offrir des cartes de fidélités et des codes de parrainages pour booster sa communication. D’où l’idée de génie de créer une application mobile à cet effet.

1. Objectifs

L'application vise à :

* Améliorer le suivi des commandes,
* Optimiser la gestion des livraisons,
* Assurer la traçabilité des ventes,
* Offrir une vue en temps réel des statistiques,
* Permettre la création de promotions, événements et mini-jeux.

1. Enjeux

Les principaux avantages de cette application incluent :

* **Amélioration de l'expérience client** : rendre le processus de commande fluide et intuitif,
* **Visibilité accrue auprès des étudiants** : via une application moderne et accessible,
* **Fidélisation des clients** : grâce aux programmes de fidélité et offres attractives

1. Porté du projet

Le projet comprend :

* Le développement d'une application web avec intégration d'un système de paiement mobile,
* Une extension possible vers une application mobile.

1. Parties prenantes

* La direction du restaurant
* Le gérant
* Les employés
* Les étudiants

1. Description fonctionnelle

Les besoins clés :

L’ application doit inclure :

* **Page d'accueil** : présentation du menu du jour, des promotions et des événements,
* **Page d'authentification** : connexion sécurisée des utilisateurs,
* **Espace étudiant** : passer des commandes, utiliser des points de fidélité, parrainer d'autres étudiants, déposer des réclamations,
* **Espace employé** : validation des commandes, mise à jour du menu, gestion des réclamations, suivi des statistiques,
* **Espace administrateur** : gestion des comptes employés et supervision des activités.

Utilisations prévues :

* L’étudiant se connecte, effectue une commande et confirme (pour approfondir en payant ). Il peut utiliser ses points de fidélités pour payer. Il peut également parrainer d’autres étudiants en demande à la personne de commander son code généré par l’application. Il peut également effectuer une réclamation.
* Les employés se connectent, valident les commandes après service, effectuent des mises à jour sur le menu, valident les réclamations, voient les stats de la semaine.
* La direction se connecte, crée des employées, voit tout dans l’application

1. Contraintes

Les contraintes techniques de ce projet incluent :

* **Multiplateforme** : l'application doit fonctionner sur tous les navigateurs web (Google Chrome, Firefox, Safari, etc.).
* **Évolutive** : l’application doit pouvoir s’adapter pour accueillir plusieurs restaurants à terme.
* **Sécurité et conformité RGPD** : respect des règles de protection des données.
* **Respect de la charte Graphique**. **(Primary color : #cfbd97, secondary color : #000000)**

##### **Etapes Techniques et Architecture**

Les étudiants devront suivre les étapes techniques suivantes pour réaliser ce projet :

1. **Planification et gestion du projet dans Jira** :
   * Créer un projet dans Jira pour suivre les tâches, les fonctionnalités, les bugs et la progression du projet.
   * Définir un backlog des tâches et user stories en lien avec les fonctionnalités à implémenter.
2. **Stockage du code sur GitHub** :
   * Créer un dépôt GitHub pour versionner le code.
   * Collaborer en équipe en utilisant les branches Git pour différentes fonctionnalités.
3. **Déploiement de la maquette sur Vercel** :
   * Créer une maquette Figma de l’application et la déployer sur **Vercel** pour permettre une démonstration rapide.
   * Vérifier le fonctionnement de l’application sur les différents navigateurs et appareils.
4. **Réalisation des tests d'application** :
   * Écrire des tests unitaires et des tests fonctionnels pour vérifier le bon fonctionnement de l'application.
   * Effectuer des tests end-to-end pour valider l'expérience utilisateur complète, de la commande à la livraison.
5. **Description de l'architecture** :
   * L'application devra adopter une **architecture MVC** (Modèle-Vue-Contrôleur) ou une architecture similaire, avec :
     + **Modèle** : gestion des données et interactions avec la base de données.
     + **Vue** : interface utilisateur visible par les étudiants, employés et administrateurs.
     + **Contrôleur** : gestion de la logique métier, reliant le modèle à la vue.
   * Utiliser des technologies web modernes (ex : **React**, **Vue.js**, ou **Angular** pour le frontend libre à vous d’utiliser la technologie avec laquelle vous vous sentez à l’aise , **php** , **Laravel** pour le backend).
6. **Mise en production** :
   * Mettre en place une base de données pour la gestion des utilisateurs, des commandes et des statistiques.

**Description technique des fonctionnalités :**

### **1. Espace Étudiant**

L'espace étudiant est conçu pour offrir une expérience utilisateur fluide et engageante, facilitant la commande, la participation aux programmes de fidélité, et l'accès aux promotions. Les étudiants peuvent également parrainer d'autres étudiants et accumuler des points de fidélité.

#### **Fonctionnalités :**

* **Inscription et Connexion** : L'étudiant peut créer un compte et se connecter avec des informations telles que l'email et le mot de passe (contenant au moins une majuscule et au moins un chiffre. ) , localisation pour les livraisons , nom , numéro de téléphone. Un lien de récupération de mot de passe est également disponible.
* **Visualisation du Menu** : Affichage du menu du jour et des promotions en cours.
* **Commande en Ligne (Système de panier )** : L'étudiant peut sélectionner des articles du menu, les ajouter au panier et finaliser la commande. Les commandes peuvent être servis sur place (Heure d’arrivée ) ou sous livraison. Le client peut laisser un commentaire une fois sa commande livre
* **Paiement en Ligne** : L'étudiant peut payer via un agrégateur de paiements (carte bancaire, mobile money, etc.). Il peut utiliser ses points de fidélité accumulés pour réduire le montant de la commande.
* **Programme de Fidélité** : L'étudiant accumule des points de fidélité à chaque commande et peut les utiliser pour des remises.
* **Parrainage** : Chaque étudiant dispose d'un code de parrainage unique qu'il peut partager avec d'autres étudiants. Lorsqu'un autre étudiant utilise ce code pour commander, le parrain accumule des points de fidélités supplémentaires.
* **Historique des Commandes** : L'étudiant peut consulter son historique de commandes, voir les détails de chaque transaction, et soumettre des réclamations si nécessaire.
* **Réclamations** : L'étudiant peut signaler un problème avec une commande et suivre l'état de sa réclamation.
* **Participation aux Jeux et Événements** : Accès à des mini-jeux et événements proposés par le restaurant pour gagner des prix ou des points supplémentaires.
* **Visualisation des 10 meilleurs clients**: Le client pourra avoir accès a la liste des 10 meilleurs clients ayants le plus de commandes et pouvant faire des filtres sur le mois , la semaine et le jour .
* **Une fenêtre informe l'utilisateur de la finalité des cookies en lui demandant son consentement à leurs utilisations.**

**2. Espace Administrateur**

L’espace administrateur est réservé à la direction du restaurant, offrant des droits de gestion complets sur l'application. Cet espace permet une supervision globale des opérations et des employés.

**Fonctionnalités :**

* **Gestion des Employés** : L'administrateur peut ajouter, modifier ou supprimer des comptes employés.
* **Gestion du Menu** : L'administrateur peut créer, modifier et supprimer les éléments du menu (plats, boissons, etc.).
* **Statistiques en Temps Réel** : L'administrateur peut consulter des statistiques sur les ventes, les commandes, et l'utilisation des programmes de fidélité et de parrainage.
* **Gestion des Promotions** : L'administrateur peut créer des offres spéciales, promotions et événements pour stimuler les ventes. [ Les offres et jeux ici seront introduites sous forme d’affiches]
* **Suivi des Réclamations** : L'administrateur peut consulter et gérer les réclamations soumises par les étudiants.
* **Gestion des Paramètres de l’Application** : L'administrateur peut configurer les paramètres généraux de l'application (heures d'ouverture, politiques, etc.).

**Liste des tâches pour l'Espace Administrateur :**

1. Créer l'interface pour la gestion des comptes employés.
2. Développer la page de gestion du menu (CRUD - Create, Read, Update, Delete).
3. Implémenter la fonctionnalité de statistiques en temps réel.
4. Mettre en place le système de gestion des promotions et événements.
5. Créer une interface de suivi des réclamations des étudiants.

**3. Espace Employé**

L'espace employé permet aux travailleurs du restaurant de gérer le flux des commandes, d'effectuer des mises à jour sur le menu, et de traiter les réclamations des étudiants.

**Fonctionnalités :**

* **Connexion et Gestion des Commandes** : Les employés se connectent et accèdent à une interface où ils peuvent consulter, préparer et valider les commandes des étudiants.
* **Mise à Jour du Menu** : Les employés peuvent effectuer des mises à jour temporaires sur le menu (exemple : plat épuisé, changement du plat du jour).
* **Validation des Réclamations** : Les employés peuvent suivre et répondre aux réclamations des étudiants.
* **Accès aux Statistiques de Vente Hebdomadaires** : Les employés peuvent consulter les statistiques sur les ventes de la semaine pour ajuster leur flux de travail.

**Liste des tâches pour l'Espace Employé :**

1. Créer la page de connexion pour les employés.
2. Développer l'interface de gestion des commandes (consultation et validation).
3. Implémenter la fonctionnalité de mise à jour du menu.
4. Mettre en place un tableau de bord pour les réclamations des étudiants.
5. Créer une page pour afficher les statistiques hebdomadaires.

**4. Espace Gérant**

Le gérant du restaurant a des droits étendus sur l’application, mais sans aller jusqu’à ceux de l’administrateur. Il supervise les opérations quotidiennes et veille à la bonne gestion du restaurant.

**Fonctionnalités :**

* **Supervision des Commandes** : Le gérant peut surveiller l'état des commandes en temps réel.
* **Création de Comptes Employés** : Le gérant peut ajouter des employés dans le système pour leur attribuer des droits d’accès à l'application.
* **Statistiques Générales** : Le gérant peut consulter des statistiques sur les ventes, les commandes, et l’utilisation des programmes de fidélité et de parrainage.
* **Gestion des Réclamations** : Le gérant peut consulter les réclamations des étudiants et valider ou rejeter les réponses proposées par les employés.

**Liste des tâches pour l'Espace Gérant :**

1. Créer l'interface de supervision des commandes en temps réel.
2. Développer la fonctionnalité de gestion des comptes employés.
3. Implémenter un tableau de bord pour les statistiques générales.
4. Mettre en place une interface pour la gestion des réclamations.

### **Modèle de Système de Fidélité et de Parrainage — Explication Technique Détailée**

#### **1. Conception et Structuration du Modèle de Fidélité et Parrainage**

##### **1.1 Objectifs du Modèle de Fidélité**

Le système de fidélité vise à encourager les utilisateurs à revenir en leur offrant des points à chaque commande passée. Ces points pourront être convertis en réductions ou récompenses lors de futures commandes. Les étudiants peuvent consulter leur solde de points à tout moment et les utiliser directement à la caisse.

##### **1.2 Objectifs du Modèle de Parrainage**

Le système de parrainage permet aux étudiants d'inviter d'autres utilisateurs à utiliser l'application via un code de parrainage. L'étudiant parrain gagne des points supplémentaires lorsque son filleul effectue une première commande avec succès.

#### **2. Étapes Techniques pour la Mise en Œuvre**

##### **2.1 Modélisation des Données**

###### **2.1.1 Base de données**

Il est essentiel de concevoir une base de données qui permettra de gérer les points de fidélité, les parrainages, ainsi que les relations entre les utilisateurs. Les tables de base incluent :

* **Utilisateurs (users)** :
  + **ID utilisateur**
  + **Nom d'utilisateur**
  + **Email**
  + **Numéro de téléphone**
  + **Points de fidélité**
  + **ID parrain (nullable)**
  + **Date d'inscription**
  + **Statut du compte (actif/inactif)**
* **Commandes (orders)** :
  + **ID commande**
  + **ID utilisateur**
  + **Montant total**
  + **Nombre de points accumulés**
  + **Date de commande**
* **Parrainage (referrals)** :
  + **ID parrain**
  + **ID filleul**
  + **Date de parrainage**
  + **État (récompense obtenue ou non)**

###### **2.1.2 Calcul des Points**

Pour les points de fidélité, un mécanisme de conversion doit être mis en place pour convertir le montant des commandes en points. Par exemple :

* 1000F dépensé = 1 point
* 15 points = 1000 de réduction

##### **2.2 Architecture Backend**

###### **2.2.1 API pour la Gestion des Points de Fidélité**

Le backend devra gérer l'attribution des points de fidélité après chaque commande. Les points seront calculés en fonction du montant de la commande et seront ajoutés au solde de l'utilisateur.

Endpoints suggérés :

* **POST /orders** : Crée une nouvelle commande et met à jour les points de fidélité.
* **GET /users/{id}/points** : Récupère le solde actuel de points d’un utilisateur.

###### **2.2.2 API pour la Gestion du Parrainage**

Le système de parrainage doit permettre aux étudiants de générer des codes de parrainage, de les partager avec d’autres utilisateurs, et de recevoir des récompenses lorsque le filleul passe une première commande.

Endpoints suggérés :

* **POST /referral/generate** : Génère un code de parrainage pour un utilisateur.
* **POST /referral/use** : Permet à un utilisateur d’utiliser un code de parrainage lors de son inscription.
* **POST /referral/reward** : Attribue des points de parrainage au parrain après la première commande réussie du filleul.

##### **2.3 Frontend — Interface Utilisateur**

###### **2.3.1 Interface Étudiant pour la Fidélité**

* **Vue Solde des Points** : Une section dans le profil de l'utilisateur où il peut voir son solde actuel de points de fidélité. Un bouton permettra d’utiliser les points accumulés pour payer une commande (à condition d'avoir atteint le seuil minimal de points).
* **Historique des Points** : Un tableau où l’étudiant peut voir les points accumulés à chaque commande passée.

###### **2.3.2 Interface Étudiant pour le Parrainage**

* **Génération de Code de Parrainage** : Un bouton permettant de générer un code de parrainage unique qui sera lié à l’utilisateur.
* **Suivi des Parrainages** : Une interface qui affiche la liste des filleuls associés à l'utilisateur ainsi que l'état de leurs premières commandes. Cela permettra de savoir si une récompense a déjà été attribuée ou non.

###### **2.3.3 Interface Admin**

L'admin doit pouvoir :

* Voir les statistiques globales sur le programme de fidélité et de parrainage.
* Valider manuellement des réclamations de points, si nécessaire.
* Gérer les paramètres du programme, comme le taux de conversion des points ou la durée de validité des points.

##### **2.4 Calcul Automatique et Gestion des Points**

###### **2.4.1 Gestion des Points lors des Commandes**

Lorsque l’étudiant passe une commande, une logique de backend doit :

* Calculer les points gagnés en fonction du montant de la commande.
* Ajouter ces points à son solde actuel.

Si l'étudiant souhaite utiliser ses points pour payer une partie de la commande, la logique doit :

* Vérifier le solde de points disponible.
* Calculer la réduction applicable sur la commande.
* Déduire les points utilisés du solde.

###### **2.4.2 Utilisation des Points et Expiration**

Les points de fidélité peuvent avoir une **date d’expiration** (par exemple, 12 mois). Le backend doit gérer l'expiration des points automatiquement en supprimant les points non utilisés après cette période.

###### **2.4.3 Gestion des Récompenses de Parrainage**

Lorsqu’un filleul passe une première commande réussie, la récompense doit être calculée et ajoutée au solde de points du parrain. Le statut du parrainage est alors mis à jour comme "récompense attribuée".

##### **2.5 Tests et Validation**

###### **2.5.1 Tests Unitaires**

* Vérification que les points sont correctement attribués après une commande.
* Test de la génération et utilisation des codes de parrainage.
* Validation des récompenses attribuées.

### **Intégration d'un Système de Paiement à Plateforme avec un Agrégateur — Explication Détaillée**

#### **1. Contexte et Objectif**

L'objectif de cette intégration est de permettre aux utilisateurs de l'application web du restaurant **ZeDuc@Space** (Mon Miam Miam) de payer leurs commandes via un **agrégateur de paiements**. L'agrégateur facilite la gestion de multiples méthodes de paiement en unifiant les interactions avec diverses institutions financières (cartes bancaires, mobile money, etc.). Cela permet au restaurant de gérer plusieurs types de transactions tout en réduisant la complexité.

#### **2. Qu'est-ce qu'un Agrégateur de Paiements ?**

Un agrégateur de paiements est un service qui regroupe plusieurs passerelles de paiement et offre une API unique pour interagir avec différentes solutions de paiement (par exemple, carte bancaire, PayPal, Mobile Money). En fonction de la région, un agrégateur de paiement peut intégrer des services tels que **CinetPay**, **Stripe**, ou **PayDunya**. L'agrégateur simplifie ainsi la gestion de plusieurs moyens de paiement en fournissant :

* Une interface unique pour différents modes de paiement.
* La gestion des transactions de bout en bout (autorisation, validation, remboursement, etc.).
* Un tableau de bord pour suivre les paiements en temps réel.

#### **3. Étapes Techniques pour l'Intégration**

##### **3.1 Étape 1 : Choix de l'Agrégateur de Paiements**

Avant de procéder à l’intégration technique, il est essentiel de choisir un agrégateur de paiement adapté aux besoins du restaurant et aux utilisateurs ciblés. Les critères à considérer incluent :

* La couverture des méthodes de paiement (cartes bancaires, Mobile Money, etc.).
* Les frais de transaction.
* La documentation technique et les APIs offertes par l’agrégateur.

Pour ce projet, des agrégateurs comme **CinetPay** (populaire en Afrique) ou **Stripe** (global) pourraient être pertinents.

##### **3.2 Étape 2 : Création d’un Compte auprès de l’Agrégateur**

Le restaurant **ZeDuc@Space** doit créer un compte commercial auprès de l'agrégateur de paiements choisi. Ce compte est nécessaire pour obtenir les **clés API** et d'autres informations de configuration pour l'intégration.

Une fois le compte validé, l'agrégateur fournit :

* **Clé publique** : utilisée pour identifier les transactions publiques côté frontend.
* **Clé privée** : utilisée pour authentifier et sécuriser les transactions côté backend

Un exemple : <https://github.com/DarylNoupik/CinetPay-Laravel>

# Délais et Équipe

Le projet **"Mon Miam Miam"** sera réalisé sur une période de **4 semaines [30 septembre - 21 Octobre]**, avec une équipe de **6 personnes par groupe**. Ce délai serré sera respecté grâce à une organisation rigoureuse, une distribution efficace des tâches, et une gestion dynamique des priorités.

Chaque membre de l'équipe aura des rôles bien définis afin de garantir une exécution rapide et fluide des différentes phases du projet.

L’équipe travaillera selon une méthodologie agile (par exemple, **Scrum**), avec des sprints quotidiens permettant de faire des ajustements en temps réel et d'assurer que chaque étape soit respectée dans les délais impartis. Des **points de contrôle réguliers** seront effectués pour suivre l’avancement des tâches et ajuster les priorités si nécessaire.

**Autre note** : Les tableaux sont dynamiquement remplis en AJAX afin d’avoir les mises a jour automatiques des tableaux .

# Budget

Une analyse budgétaire du projet sera effectuée et proposée par l’équipe projet .

# Livrables

### **1. Livrables Techniques**

Ces éléments sont directement liés à la conception, au développement, aux tests et au déploiement de l'application web.

#### **A. Développement et Fonctionnalités**

1. **Code source** :
   * Code complet de l’application (frontend et backend) hébergé sur **GitHub**.
   * Structuration du projet avec une séparation claire entre le frontend (interface utilisateur) et le backend (API, logique métier).
2. **Interfaces Utilisateurs Figma (UI)** :
   * **Page d’accueil** (accès rapide aux informations : menu du jour, promotions, événements).
   * **Page d’authentification** (login et inscription des utilisateurs).
   * **Espace étudiant** (commande, points de fidélité, parrainage, réclamations).
   * **Espace employé** (validation des commandes, mise à jour du menu, traitement des réclamations).
   * **Espace administrateur** (gestion des employés, suivi des statistiques).
   * **Espace gérant** (création d’employés, visualisation des données globales).
3. **Documentation des API Backend** :
   * API RESTful permettant la gestion des commandes, des utilisateurs, du programme de fidélité, et du parrainage.
   * **Gestion des paiements** : intégration d’un système de paiement via un agrégateur (Mobile Money ou autres).[ Bonus , cette fonctionnalité n’est pas obligatoire ]
   * **Base de données** : Modélisation de la base de données pour les utilisateurs, les commandes, les points de fidélité, et les parrainages.

#### **B. Infrastructure et Déploiement**

1. **Maquette interactive** :
   * **Déploiement sur Vercel** de la maquette de l’application pour une présentation des interfaces et des parcours utilisateurs.
2. **Environnement de production** :
   * **Procédure de déploiement backend** (API) sur une plateforme cloud
   * Etape d’intégration d’une **CI/CD pipeline** (par exemple, avec GitHub Actions) pour automatiser les déploiements lors de mises à jour du code.

#### **C. Tests et Validation**

1. **Tests unitaires et fonctionnels** :
   * Tests automatisés couvrant les fonctionnalités critiques (connexion, commande, paiement, etc.).
   * **Rapport de test** avec couverture des fonctionnalités principales.

### **2. Livrables Non Techniques**

Ces éléments assurent la gestion, la documentation et le bon déroulement du projet.

#### **A. Documentation**

1. **Documentation technique** :
   * **Description de l’architecture** du projet : explication du choix des technologies (backend, frontend, base de données), des modules et services, ainsi que de la structure des API.
   * **Guide d’installation** et d'utilisation : explication des étapes pour installer et exécuter le projet en local et en production.
   * **Schémas de la base de données** : représentation des tables, relations, et données nécessaires.
   * **Charte graphique de l’application (Police** – Couleurs)
2. **Manuel d’utilisation** :
   * **Guide utilisateur** détaillé pour chaque rôle (étudiant, employé, administrateur, gérant), expliquant les fonctionnalités disponibles dans chaque espace.

#### **B. Gestion de projet**

1. **Planning sur Jira** :
   * Suivi des tâches avec **Jira** pour assurer le bon déroulement du projet, incluant les étapes de développement, tests, et déploiement.
   * **Rapport d’avancement** des sprints (ou étapes) pour le suivi des deadlines.
2. **Rapports de réunion** :
   * **Comptes-rendus** des réunions d'équipe et des points d’avancement pour garder une trace des décisions prises et ajuster les priorités.

#### **C. Livrables relatifs à la gestion des risques**

1. **Plan de gestion des risques** :
   * Identification des risques liés au projet (ex. délais, bugs critiques, pannes serveur) et des mesures d’atténuation.

#### **D. Validation finale**

1. **Version test** :
   * Version préproduction de l'application disponible pour un groupe restreint d’utilisateurs (bêta-test) afin de recueillir des retours avant la version finale.
2. Analyse Budgétaire du projet
3. Les mentions légales et tout autre aspects Juridiques associes

RESSOURCES :

**Inspirations pour la maquette Figma :**

* <https://dribbble.com/search/restaurant>
* <https://dribbble.com/search/restaurant>

**Ressources techniques :**

* https://marcdezordo.me/les-differences-entre-zoning-wireframe-mockup-et-prototype/
* http://php.net/manual/fr/
* https://validator.w3.org/
* https://jigsaw.w3.org/css-validator/
* https://jquery.com/
* https://datatables.net/
* https://nodejs.org/en/
* https://www.frugalprototype.com/developpez-propre-api-node-js-express/
* <https://www.getpostman.com/>
* <https://www.atlassian.com/fr/software/jira>
* <https://grafikart.fr/formations/laravel>
* <https://www.youtube.com/watch?v=oGPIks7Ggh8&list=PLDc9bt_00KcJvM4-KJF8OPadbOod_G7Wf>
* <https://www.subdelirium.com/generateur-de-mentions-legales/>
* <https://www.subdelirium.com/generateur-de-mentions-legales/>