# Sommaire

Table des matières

**Tableau de Synthèse2**

**Attestation de Travail3**

**Description succinte des projets4**

Projet 1 : 1ClickAllEat5

Projet 2 : LDAP - Telora\*

Projet 3 : Portfolio\*

**Veille technologique10**

**Annexe 11**

Projet 1 : 1ClickAllEat12

Cahier des Charges12

UML16

Extrait de code24

gantt32

Méthode merise33

Documentation Utilisateur38

Projet 2 : LDAP - Telora48

Tapez le titre du chapitre (niveau 3)6

Annexe veille\*\*

1. Tableau de Synthèse

(a insérer une fois imprimer )

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Gérer le patrimoine informatique** | **Répondre aux incidents et aux demandes d’assistance et d’évolution** | **Développer la présence en ligne de l’organisation** | **Travailler en mode projet** | **Mettre à disposition des utilisateurs un service informatique** | **Organiser son développement professionnel** |
| **Portfolio** |  |  | X | X |  | X |
| **1ClickAllEat** |  |  | X | X | X | X |
| **Logiciel LDAP** | X | X |  | X | X | X |

1. Attestation de Travail

A insérer : attestation fournie par l’entreprise ou le tuteur de stage/alternance si applicable.

1. Descriptions Succinctes des Projets

Projet 1 : 1CLickAllEat

Contexte : Projet réalisé en autonomie dans le cadre du BTS SIO SLAM, Pour répondre à un besoin concret : fluidifier la gestion des réservations et des commandes dans la restauration, tout en offrant une expérience utilisateur moderne, rapide et sécurisée.

Partenaire : Aucun (projet individuel)

Environnement technique :

* + - Front-end : Blade (Laravel), Bootstrap 5, HTML5, CSS3, responsive design
    - Back-end : Laravel 10 (PHP 8), architecture MVC
    - Base de données : SQLite (développement),

MySQL/PhpMyAdmin (production)

* + - Outils : Breeze (authentification), Faker (données de test), GitHub, VS Code, hébergement HTTPS

Liste des tâches :

* + - Analyse du besoin, rédaction du cahier des charges et du diagramme de gantt
    - Modélisation de la base de données et création des migration Laravel
    - Développement des fonctionnalités principal : réservation, commande, gestion des utilisateurs, gestion des restaurants/tables/menus
    - Implémentation de l’authentification sécurisée multi-profils (clients, restaurateurs)
    - Développement d’un tableau de bord pour restaurateur (statistique, gestion des réservations et commandes)
    - Tests unitaires, validation des formulaires et gestion des erreurs
    - Déploiement sur serveur distant, configuration du HTTPS et optimisation de la sécurité
    - Rédaction de la documentation utilisateur et technique

Difficultés rencontrées / solutions :

* + - Découverte et prise en main de Laravel (documentation officielle, tutoriels, forums)
    - Gestion des droits et de la sécurité (middlewares, tests, validation)
    - Déploiement et configuration serveur (documentation OVH, Let’s Encrypt, adaptation .env)
    - Pas de maquette initiale, adaptation de templates Bootstrap (recherche d’exemples, tests responsive)

Compétences couvertes :

* + - Développement de composants d’interface
    - Création d’interfaces utilisateurs/administrateur
    - Optimisation de l’expérience utilisateur
    - Intégration de services externes
    - Architecture de base de données relationnelle
    - Optimisation des requêtes
    - Administration du patrimoine informatique
    - Protection des informations sensibles
    - Gestion des autorisations
    - Documentation des processus (UML, MERISE)

Compétences développées :

* + - Maîtrise du framework Laravel
    - Gestion autonome d’un projet complexe
    - Mise en place d’interfaces par rôle
    - Sécurisation des données utilisateurs
    - Capacité de mise en ligne et de sécurisation une fois le projet en production

Bilan personnel/Perspective d’amélioration :

* + - Fier d’avoir mené ce projet seul, en développant des fonctionnalités avancées et en progressant énormément sur Laravel et l’écosystème PHP.
    - Gain d’autonomie, de rigueur, et de capacité à résoudre des problèmes complexes.
    - Pour la suite : intégrer un paiement en ligne, notifications avancées, API mobile, personnalisation des pages par restaurant.

Projet 2 : Portfolio Web

Contexte : Dans le cadre de mon BTS SIO option SLAM, j’ai réalisé un site portfolio afin de valoriser mes compétences, mes projets et ma démarche de veille technologique. Ce projet est une initiative personnelle, sans partenaire, qui répond à la fois à un besoin professionnel (présentation lors de recherches de stage ou d’emploi) et pédagogique (synthèse de mes acquis techniques et méthodologiques).

Environnement technique :

* + - Langages : PHP, HTML5, CSS3
    - Frameworks & outils : Bootstrap 5 (pour le responsive et les composants UI), XAMPP (serveur local), JavaScript (pour les interactions dynamiques)
    - Gestion de versions : Github, Gitdesktop
    - Organisation : Arborescence MVC simplifiée, séparation du code, utilisation de includes pour le header/footer

Objectifs et fonctionnalités principales :

* + - Présenter de façon claire et attractive mes projets, compétences et expériences
    - Mettre en avant la veille technologique et mes outils de suivi
    - Proposer une navigation fluide, moderne et responsive, compatible tous supports
    - Illustrer l’évolution de mes compétences à travers des modals détaillés pour chaque projet

Démarche et tâches réalisées :

* + - Analyse des portfolios concurrents pour définir les attentes en matière de design et d’ergonomie
    - Création de la structure HTML et mise en place du routage via PHP
    - Intégration de Bootstrap pour accélérer le développement et garantir la responsivité
    - Développement de modals personnalisés pour la section veille et les projets
    - Mise en place d’un formulaire de contact sécurisé (validation des champs, protection contre les injections)
    - Ajout d’effets visuels et d’animations CSS pour améliorer l’expérience utilisateur
    - Tests sur différents navigateurs et appareils pour assurer la compatibilité
    - Rédaction de la documentation utilisateur et technique

Difficultés rencontrées / solutions :

* + - Conflits entre Bootstrap et styles personnalisés : Résolus par l’utilisation de classes spécifiques et de la propriété

**!important en CSS**.

* + - Gestion des modals et de l’accessibilité : Nécessité d’adapter le code pour garantir un affichage correct et une navigation clavier.

Bilan personnel :

* + - J’ai considérablement renforcé ma rigueur en développement front-end, en apprenant à structurer mon code et à anticiper les problèmes d’ergonomie.
    - J’ai découvert l’importance du responsive design et de l’accessibilité, en testant mon site sur mobile et en optimisant la navigation pour tous les utilisateurs.
    - Ce projet m’a permis de prendre conscience de l’importance de la veille technologique, que j’ai intégrée directement dans le site via une section dédiée et des modals interactifs.
    - Enfin, j’ai appris à documenter mon travail, à présenter mes réalisations de façon professionnelle, et à utiliser mon portfolio comme outil de communication lors de mes démarches de stage ou d’emploi.

1. Veille Technologique

Outils de veille utilisés :

* + - Feedly (agrégateur de flux RSS)
    - LinkedIn, Twitter (veille réseaux sociaux professionnels)
    - Google Alerts
    - Forums : StackOverflow, OpenClassrooms

Sujets de veille suivis :

* + PHP : nouveautés (PHP 8.3), sécurité (CVE-2024-4577), bonnes pratiques, frameworks (Laravel)
    - Sécurité web : failles récentes, bonnes pratiques, outils d’audit
  + JavaScript : nouveautés ES2024, pipeline operator, Temporal API, sécurité XSS
    - Sécurité web : failles récentes, bonnes pratiques, outils d’audit
    - Tendances développement web : frameworks, accessibilité, performance

1. Annexes

Annexes Projets

* + 1CLickAllEat
    - Cahier des charges 1ClickAllEat (p.12 à 15)
    - Diagrammes UML (p.16 à 23)
    - Extraits de code commentés (p.24 à 31)
    - Diagramme de gantt (p.32)
    - Méthode merise (p.33 à 36)
    - Documentation utilisateur (p.37 à 46)
  + Ldap - Telora
  + Portfolio

Annexes Veille

* + - Copies d’articles lus (PDF, captures)
    - Notes personnelles de synthèse

**Annexe 2 : 1ClickAllEat**

**Cahier des Charges - Projet 1ClickAllEat**

**1. Contexte**

La startup 1ClickAllEat nous mandate pour développer une application de réservation de restaurant au préalable. Cette solution vise à simplifier et rendre plus accessibles les restaurants traditionnels qui sont concurrencés par les fast-foods.

**2. Objectif Principal**

Proposer une solution optimisée pour plusieurs restaurants qui proposent des menus différents. L'application permettra d'optimiser les échanges entre les différents acteurs du restaurant (administration, service, cuisine, clients).

**3. Public Cible**

Restaurateurs et leur personnel

**4. Expression des Besoins**

L'application 1ClickAllEat doit offrir une gestion complète et optimisée des restaurants, permettant de suivre en temps réel les menus, les commandes et les réservations. L'ensemble des opérations sera parfaitement synchronisé avec les horaires d'ouverture, assurant une expérience fluide et sur-mesure pour chaque client.

La plateforme fournira aux restaurateurs les outils nécessaires à l'ajustement des offres, à la gestion des réservations et au suivi des commandes, accessibles en ligne pour un pilotage efficace des établissements à tout moment.

**5. Fonctionnalités**

**5.1. Fonctionnalités MVP (Minimum Viable Product)**

* **Gestion des restaurants**
  + Création et paramétrage des profils de restaurants
  + Configuration des horaires d'ouverture
  + Gestion des tables et capacités
* **Authentification des utilisateurs**
  + Système d'inscription/connexion sécurisé
  + Récupération de mot de passe
  + Différents niveaux d'accès selon les rôles
* **Accès spécifique restaurant**
  + Interface dédiée aux restaurateurs
  + **Tableau de bord de gestion**
  + Statistiques de base sur l'activité
* **Accès spécifique client**
  + Interface utilisateur intuitive
  + Recherche et filtrage des restaurants
  + Historique des commandes/réservations
* **Gestion de la carte/des menus**
  + Classification des articles au menu par catégorie
  + Ajout/modification/suppression d'articles
  + Gestion des prix et disponibilités
* **Gestion des commandes/réservations**
  + Création et suivi des réservations
  + Gestion des statuts des commandes
  + Notifications aux clients et restaurateurs

**5.2. Fonctionnalités MLP (Minimum Lovable Product)**

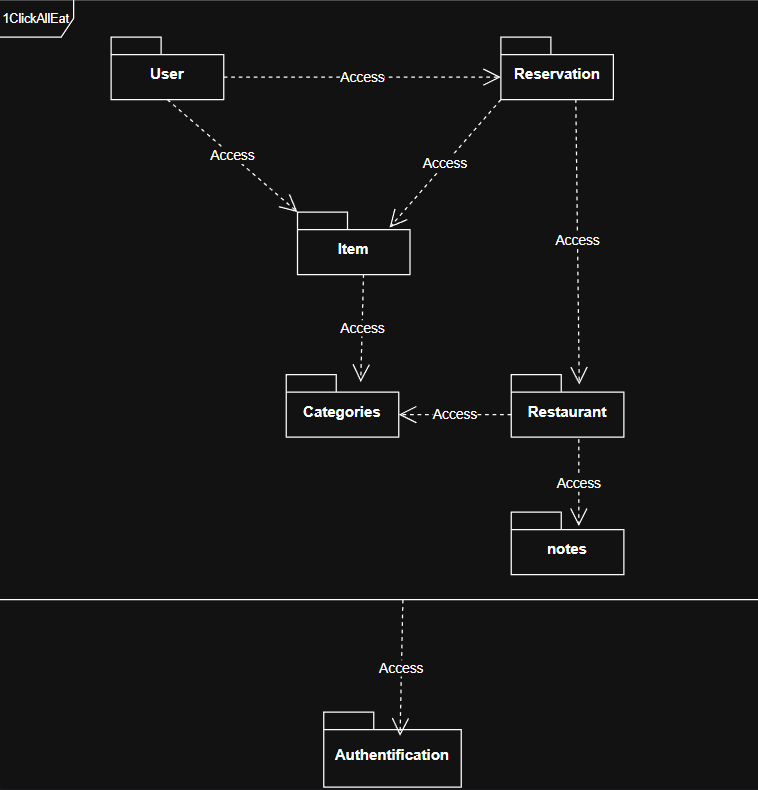
* **Génération de QR code**
  + Lien d'accès à la commande pour chaque carte de restaurant
  + QR code dynamique et personnalisable
* **Responsive**
  + Adaptation à tous les formats d'écran
  + Expérience utilisateur optimisée sur mobile, tablette et desktop
* **Paramétrage de la charte graphique**
  + Personnalisation des couleurs et logo du restaurant
  + Adaptation visuelle de la carte selon l'identité du restaurant
* **Paiement en ligne via API Stripe**
  + Intégration complète avec le package Laravel Cashier
  + Gestion sécurisée des transactions
  + Suivi des paiements et remboursements

**6. Contraintes Techniques**

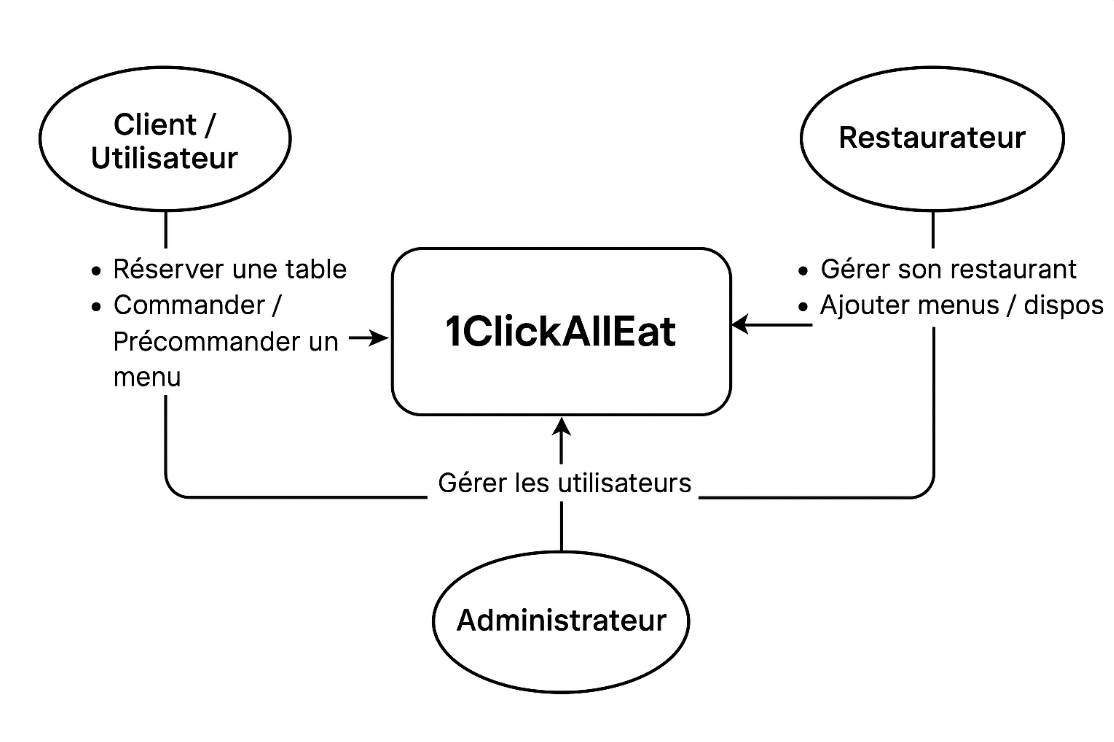
* **Framework de développement**
  + Laravel (Back-end)
  + Interface utilisateur moderne et intuitive
* **Déploiement**
  + Sur un nom de domaine dédié
  + Hébergement sécurisé et performant
* **Intégration continue**
  + Automatisation des migrations de base de données
  + Gestion du vidage des caches de Laravel
  + Optimisation des fichiers pour l'environnement de production
* **Template**
  + Design responsive et moderne
  + Adaptable à l'identité visuelle des différents restaurants
* **Sécurité**
  + Certificat SSL (HTTPS) pour sécuriser les communications
  + Protection des données personnelles
* **Tests unitaires**
  + Mise en place de tests automatisés
  + Code qui teste le code

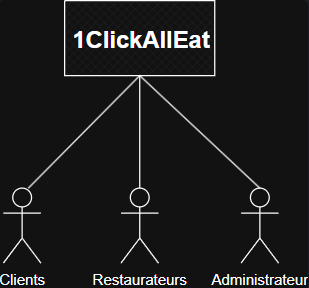
Diagramme UML :

* + Diagramme de package

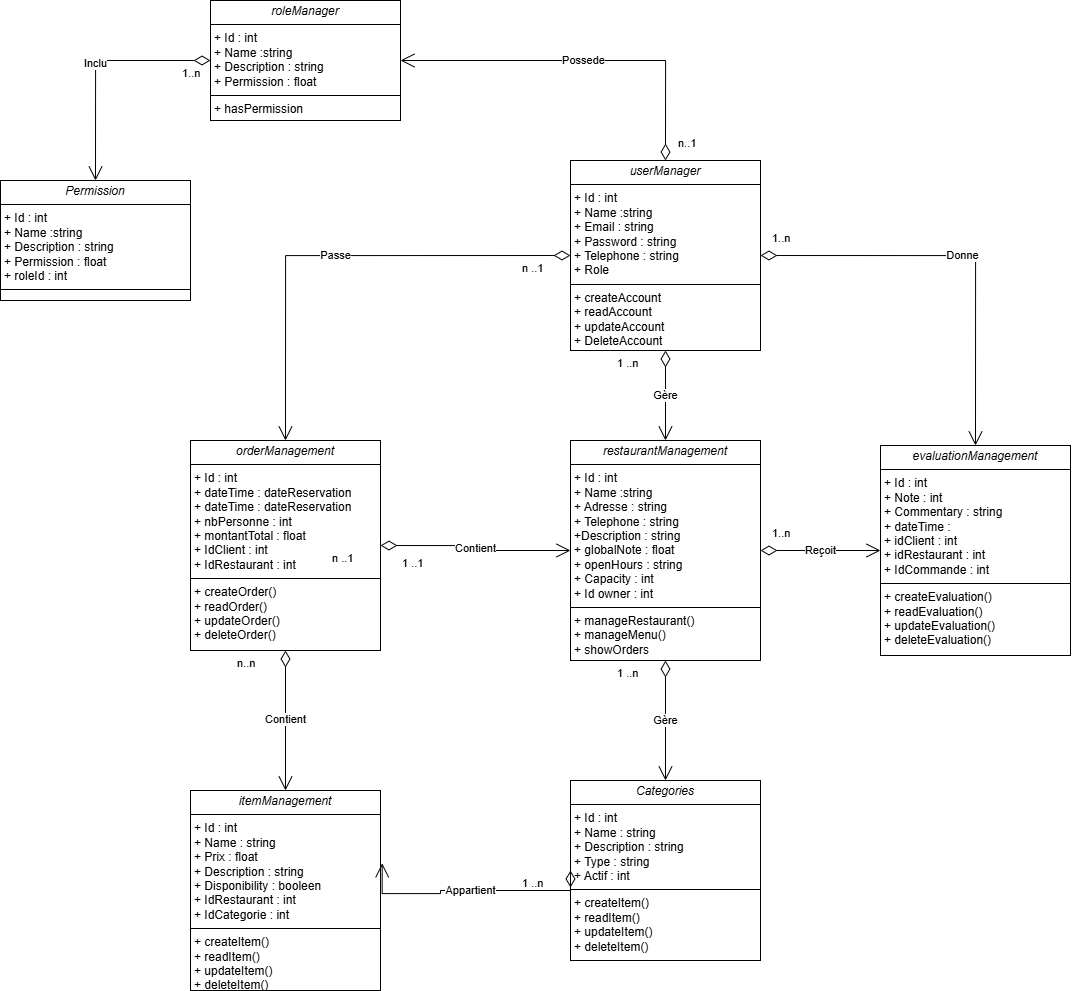


* + Diagramme de contexte

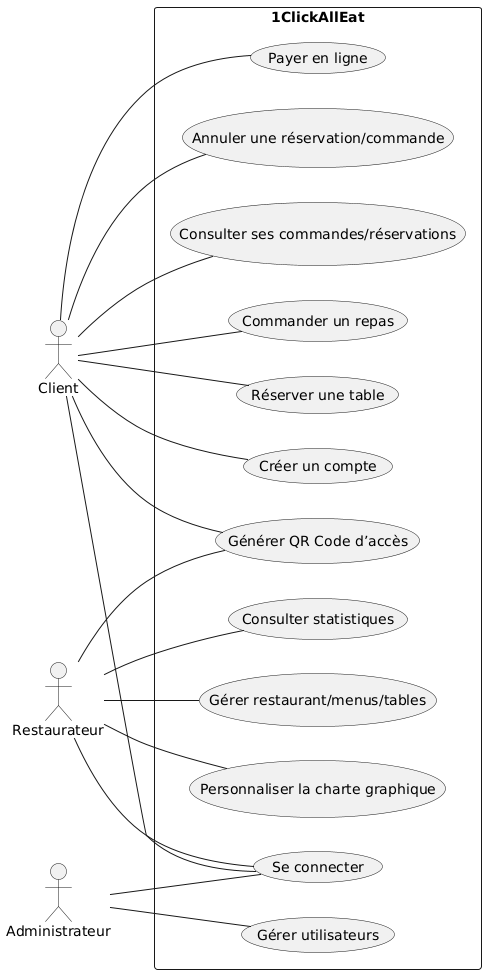




* + Diagramme de classe



* + Use\_case



Plan de Test Qualité – 1ClickAllEat

Objectif

Fournir un cadre structuré pour valider que l’application 1ClickAllEat répond à toutes les exigences fonctionnelles (MVP, MLP), techniques et de qualité : gestion des restaurants, authentification, accès différencié, gestion des menus/commandes, responsive, personnalisation graphique, sécurité, CI/CD, etc.

Périmètre du test

Fonctionnalités MVP

* Gestion des restaurants (création, modification, suppression)
* Authentification et gestion des utilisateurs (clients, restaurateurs)
* Accès différencié (dashboard restaurateur, espace client)
* Gestion des menus/cartes (catégorisation, affichage)
* Gestion des commandes (création, suivi, annulation)

Fonctionnalités MLP

* Génération et affichage du QR Code pour accès rapide
* Design responsive (mobile, tablette, desktop)
* Personnalisation graphique (charte, couleurs, logos)
* Paiement en ligne (API Stripe)

Contraintes techniques

* Laravel (PHP 8+)
* Déploiement HTTPS (certificat SSL)
* Intégration continue (migrations, cache, optimisation)
* Tests unitaires/fonctionnels automatisés

Méthodologie

* Tests manuels et automatisés : PHPUnit,
* Tests fonctionnels : Vérification de chaque exigence du cahier des charges
* Tests d’intégration : Vérification de la communication entre modules
* Tests de performance : Simulation de charge, temps de réponse
* Tests de sécurité : SSL, authentification, gestion des rôles
* Tests de compatibilité/responsive : Desktop, tablette, mobile, principaux navigateurs
* Environnement de préproduction : Identique à la production

Tableau du plan de test

| Périmètre testé | Date | Description du test | Résultat attendu | Résultat obtenu | Réussite/Échec | Testé par |
| --- | --- | --- | --- | --- | --- | --- |
| Authentification utilisateur | 2025-04-10 | Connexion/déconnexion (client, restaurateur, admin) | Accès sécurisé, redirection correcte, session isolée | À renseigner | À renseigner | QA |
| Gestion des restaurants | 2025-04-11 | CRUD restaurant via dashboard restaurateur | Opérations sans erreur, affichage immédiat des changements | À renseigner | À renseigner | QA |
| Accès restaurateur | 2025-04-12 | Dashboard, gestion menus, commandes, réservations | Accès réservé, outils visibles uniquement pour restaurateur | À renseigner | À renseigner | QA |
| Accès client | 2025-04-12 | Consultation menus, réservation, commande | Accès restreint, affichage personnalisé selon le restaurant | À renseigner | À renseigner | QA |
| Gestion menus/cartes | 2025-04-13 | CRUD catégories/articles, affichage dynamique | Modifications visibles, catégorisation correcte | À renseigner | À renseigner | QA |
| Gestion des commandes | 2025-04-14 | Passer/annuler commande, suivi en temps réel | Commande enregistrée, notification cuisine, historique mis à jour | À renseigner | À renseigner | QA |
| Génération QR Code | 2025-04-15 | Génération, scan, redirection vers menu/commande | QR code lisible, redirection correcte | À renseigner | À renseigner | QA |
| Responsive design | 2025-04-16 | Tests sur desktop, tablette, mobile, Chrome, Firefox, Safari | Interface cohérente, aucune perte de fonctionnalité | À renseigner | À renseigner | QA |
| Personnalisation graphique | 2025-04-17 | Modification couleurs, logos, polices via interface | Rendu conforme à la charte définie | À renseigner | À renseigner | QA |
| Paiement en ligne (Stripe) | 2025-04-18 | Transaction test, validation, génération reçu | Paiement validé, reçu généré, sécurité respectée | À renseigner | À renseigner | QA |
| CI/CD et déploiement | 2025-04-19 | Migrations, cache, optimisation lors d’un déploiement | Scripts CI déclenchés, déploiement sans interruption, rollback possible | À renseigner | À renseigner | DevOps |
| Sécurité SSL/API | 2025-04-20 | Vérification HTTPS, sécurité Stripe, rôles | Aucune faille détectée, accès sécurisé, logs d’accès complets | À renseigner | À renseigner | Sécurité |

Exécution, reporting et validation

* Les anomalies sont tracées dans l’outil de suivi (GitHub Issues…).
* Un rapport de test est généré à chaque fin de cycle.
* La mise en production n’est validée que si tous les tests critiques sont passés et les anomalies bloquantes corrigées.

Conclusion

Ce plan de test garantit la conformité de l’application 1ClickAllEat avec les attentes fonctionnelles et techniques du projet. Il sera mis à jour à chaque évolution majeure du produit ou retour utilisateur.

* + Extrait de code

1. Réservation d’une table

php

CopyInsert

// Extrait de ReservationController.php

public function store(Request $request, Restaurant $restaurant)

{

$validated = $request->validate([

'table\_id' => ['required', 'exists:restaurant\_tables,id'],

'date\_reservation' => ['required', 'date', 'after:now'],

'heure\_reservation' => ['required', 'date\_format:H:i'],

]);

$table = RestaurantTable::findOrFail($validated['table\_id']);

if (!Auth::check()) {

return redirect()->back()->with('error', 'Vous devez être connecté pour réserver une table.');

}

$dateTime = $validated['date\_reservation'] . ' ' . $validated['heure\_reservation'];

$exists = $table->reservations()

->where('date\_reservation', $dateTime)

->whereIn('status', ['pending', 'confirmed'])

->exists();

if ($exists) {

return redirect()->back()->with('error', 'Cette table est déjà réservée pour ce créneau.');

}

$reservation = Reservation::create([

'user\_id' => Auth::id(),

'restaurant\_id' => $restaurant->id,

'table\_id' => $table->id,

'date\_reservation' => $dateTime,

'status' => 'pending',

]);

// Création automatique d'une commande liée à la réservation

$order = new \App\Models\FoodOrder([

'client\_id' => Auth::id(),

'restaurant\_id' => $restaurant->id,

'status' => 'pending',

'total\_price' => 0

]);

$order->save();

}

**Explication détaillée**  
Cette méthode permet à un utilisateur connecté de réserver une table :

* Elle valide que la table existe et que la date/heure sont correctes.
* Elle vérifie que l’utilisateur est bien connecté.
* Elle contrôle que la table n’est pas déjà réservée à ce créneau.
* Si tout est OK, elle crée la réservation et génère automatiquement une commande vide associée, facilitant le workflow côté client et restaurateur.

**2. Affichage différencié des commandes et réservations (client vs restaurateur)**

php

CopyInsert

// Extrait de FoodOrderController.php

public function index()

{

if (Auth::user()->role->name === 'client') {

$orders = Auth::user()->clientFoodOrders()->with(['restaurant', 'items'])->latest()->get();

$reservations = \App\Models\Reservation::with(['restaurant', 'table'])

->where('user\_id', Auth::id())

->orderBy('date\_reservation', 'asc')

->get();

return view('orders.index', compact('orders', 'reservations'));

} else {

$orders = Auth::user()->restaurantFoodOrders()->with(['client', 'items'])->latest()->get();

return view('orders.index', compact('orders'));

}

}

**Explication détaillée**  
Cette méthode affiche les commandes et réservations dans l’espace utilisateur :

* Si l’utilisateur est un **client** : il voit ses propres commandes et réservations.
* Si c’est un **restaurateur** : il voit toutes les commandes passées dans son restaurant.
* Cela permet une expérience personnalisée et sécurisée selon le rôle, en limitant l’accès aux bonnes informations.

3. Passage d’une commande (création d’une commande à partir du panier)

php

CopyInsert

// Extrait de FoodOrderController.php

public function store(Request $request)

{

$validated = $request->validate([

'restaurant\_id' => ['required', 'exists:restaurants,id'],

'items' => ['required', 'array'],

]);

$itemIds = array\_keys($validated['items']);

$items = Item::findMany($itemIds);

$selectedItems = [];

foreach ($items as $item) {

$qty = intval($validated['items'][$item->id]['quantity'] ?? 0);

if ($qty > 0) {

$selectedItems[] = [

'id' => $item->id,

'quantity' => $qty,

'price' => $item->effective\_price,

];

}

}

if (count($selectedItems) === 0) {

return back()->with('error', 'Veuillez sélectionner au moins un plat.');

}

$restaurant = Restaurant::findOrFail($validated['restaurant\_id']);

// Création de la commande

$order = FoodOrder::create([

'client\_id' => Auth::id(),

'restaurant\_id' => $restaurant->id,

'status' => 'pending',

'total\_price' => array\_sum(array\_map(fn($i) => $i['quantity'] \* $i['price'], $selectedItems)),

]);

// Ajout des plats à la commande (OrderItem)

foreach ($selectedItems as $item) {

$order->items()->create([

'item\_id' => $item['id'],

'quantity' => $item['quantity'],

'price' => $item['price'],

]);

}

return redirect()->route('orders.index')->with('success', 'Commande enregistrée !');

}

**Explication détaillée**

* Cette méthode gère la validation du panier et la création d’une commande.
* Elle vérifie que le panier contient bien des plats, puis crée la commande avec le total calculé.
* Chaque plat sélectionné est ajouté à la commande via la relation items().
* Cela garantit la cohérence des données et permet un suivi précis des commandes côté cuisine.

**4. Annulation d’une réservation**

php

CopyInsert

// Extrait de ReservationController.php

public function cancel(Reservation $reservation)

{

// Vérifie que seul le propriétaire de la réservation peut l’annuler

if ($reservation->user\_id !== Auth::id()) {

abort(403);

}

$reservation->status = 'cancelled';

$reservation->save();

return redirect()->back()->with('success', 'Réservation annulée.');

}

**Explication détaillée**

* Cette méthode permet à un utilisateur d’annuler sa propre réservation.
* Elle vérifie l’autorisation (propriétaire uniquement).
* Elle change le statut en “cancelled” et sauvegarde la modification.
* Cela protège la logique métier et évite les suppressions accidentelles ou malveillantes.

**5. Ajout d’un plat au panier (extrait simplifié)**

php

CopyInsert

// Extrait de CartController.php

public function add(Request $request, $itemId)

{

$item = Item::findOrFail($itemId);

$cart = session()->get('cart', []);

$cart[$itemId] = [

'name' => $item->name,

'quantity' => ($cart[$itemId]['quantity'] ?? 0) + 1,

'price' => $item->effective\_price,

];

session(['cart' => $cart]);

return back()->with('success', 'Plat ajouté au panier.');

}

**Explication détaillée**

* Cette méthode ajoute un plat au panier stocké en session.
* Si le plat existe déjà, on incrémente la quantité.
* Le panier est persistant tant que la session de l’utilisateur reste active.
* Cela permet une expérience utilisateur fluide et rapide, sans base de données pour le panier temporaire.
  + Diagramme de gantt

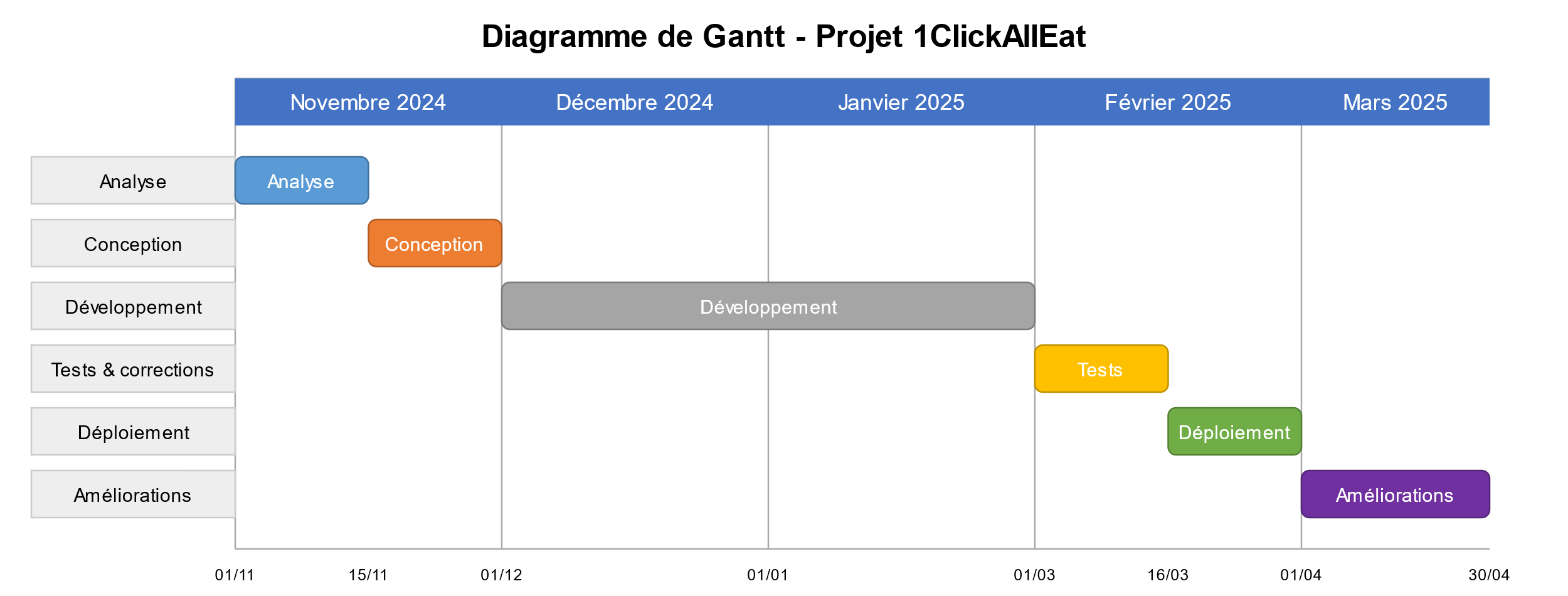
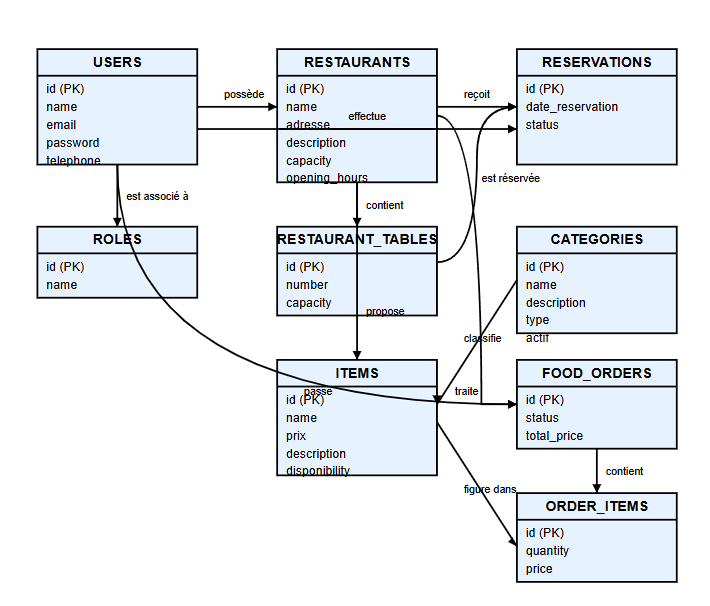


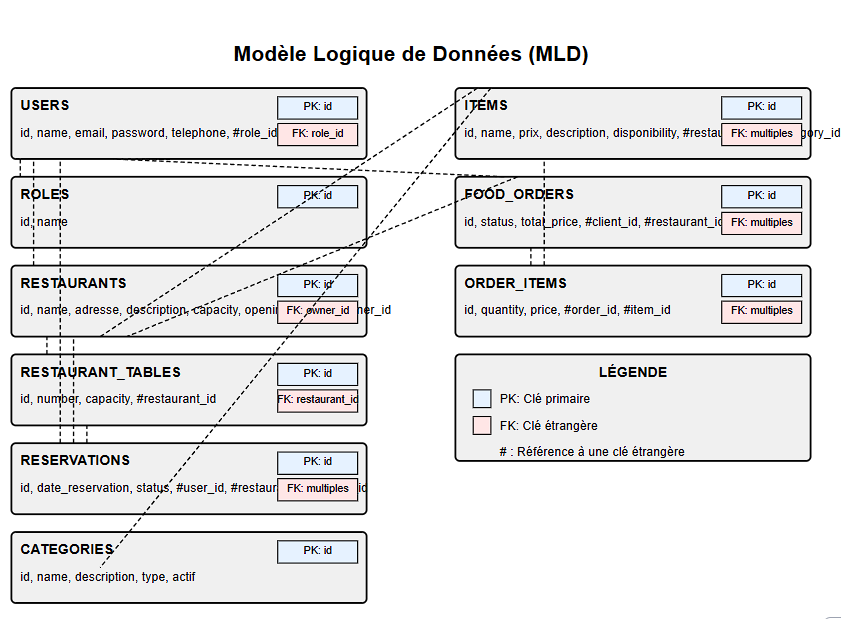
Diagramme Merise :

Dictionnaire de données : (a inprimer sur 3 page)

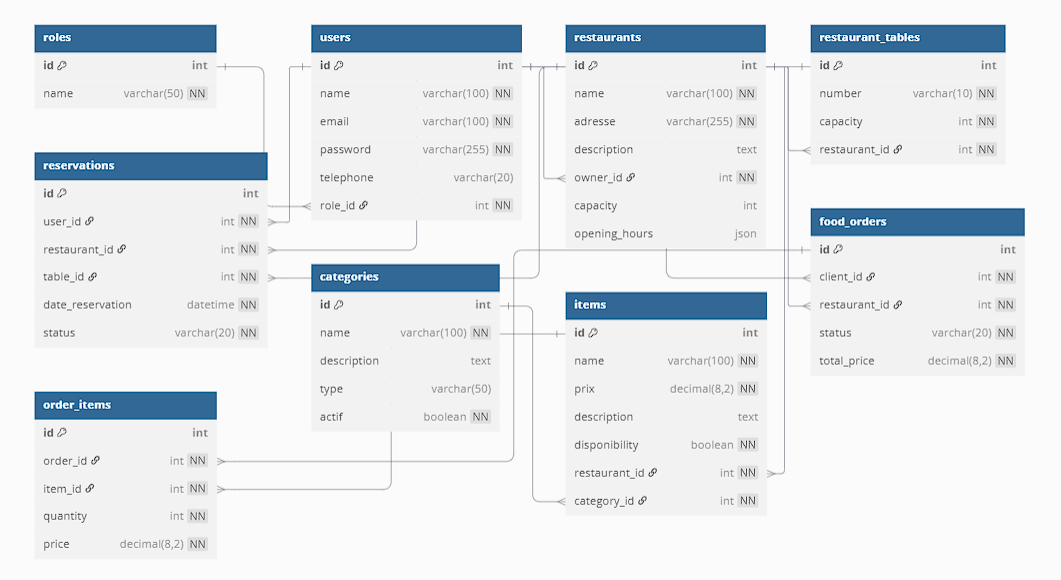
* + MCD (Modèle conceptuel de données) :



* + MLD (Modèle logique de données) :



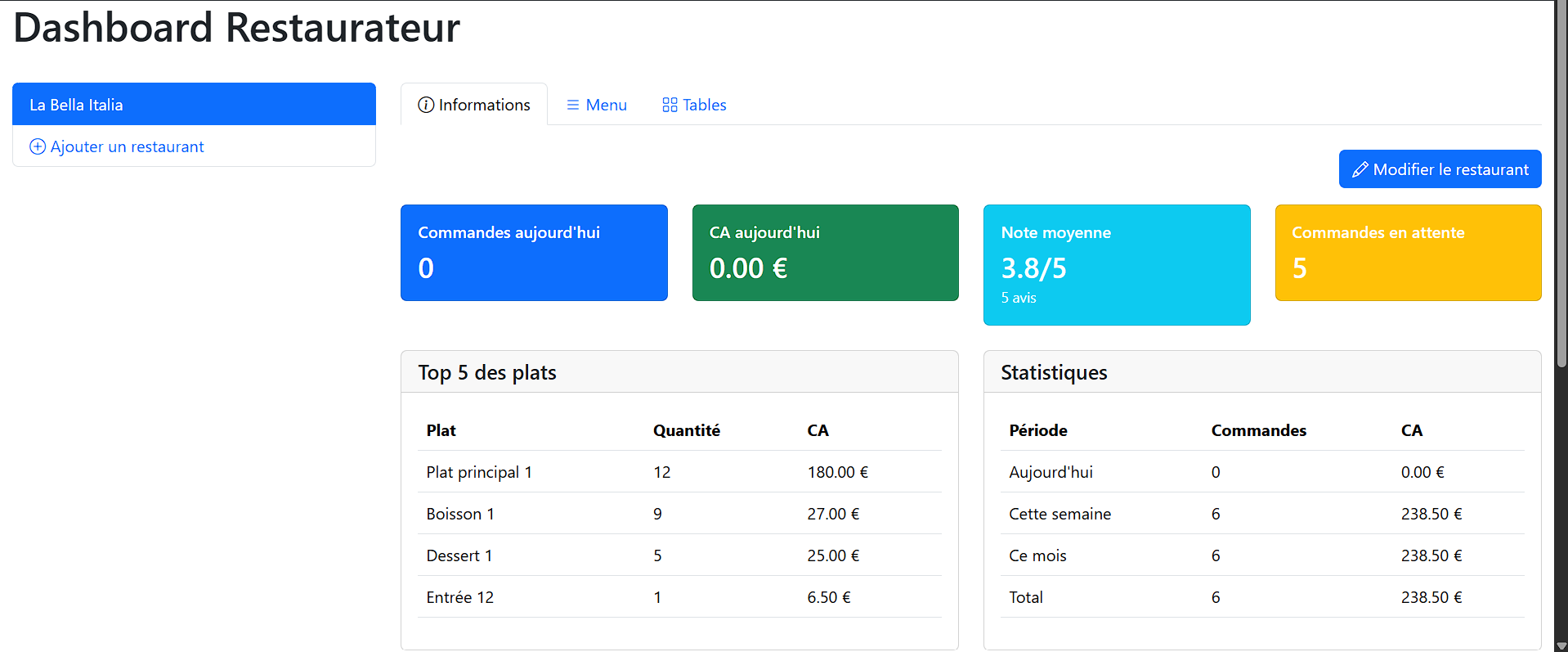
* + - MPD (Modele physique de données) :



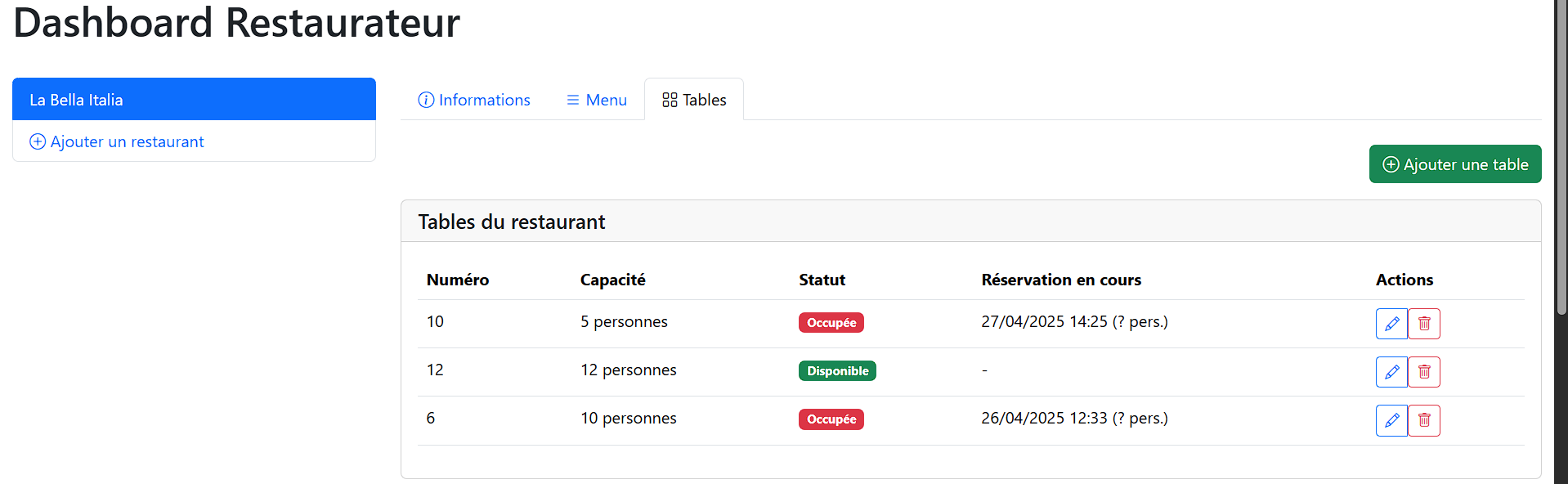
**Documentation utilisateur complète de 1ClickAllEat**

**Section 1: Interface Restaurateur**

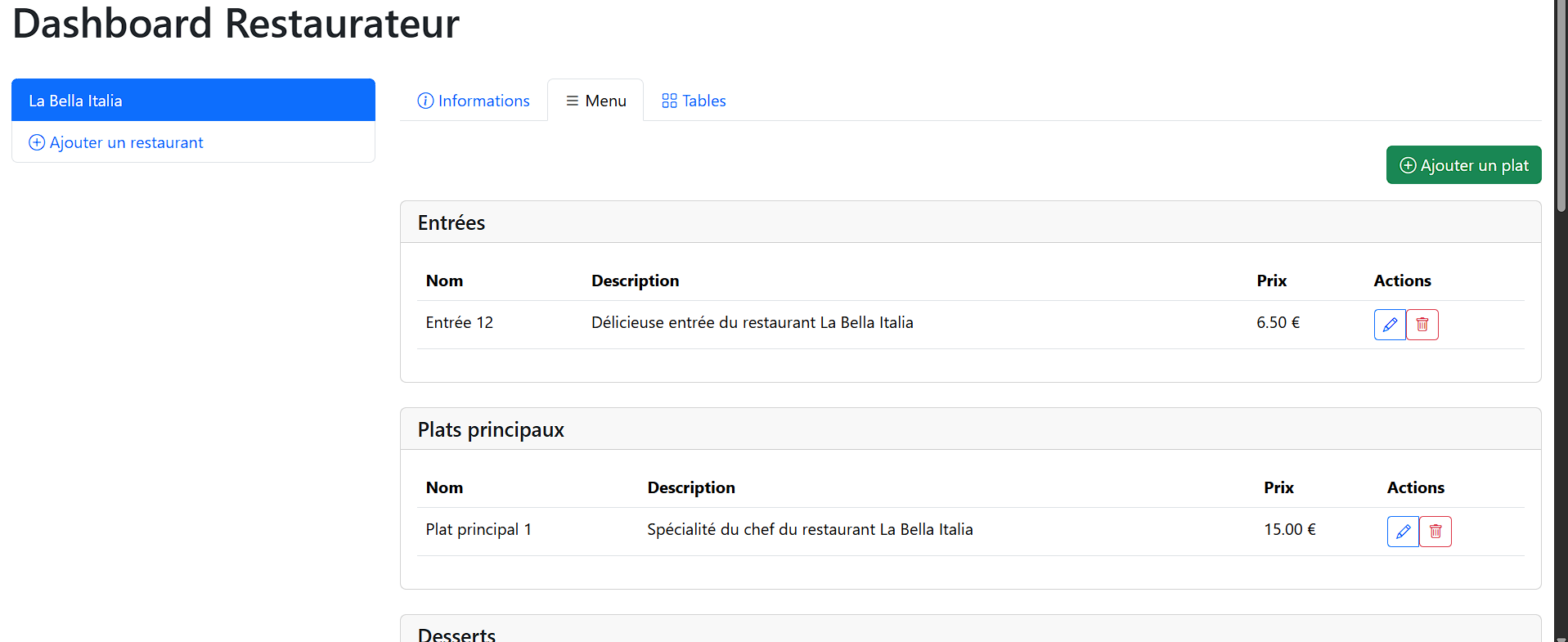
1. Dashboard principal



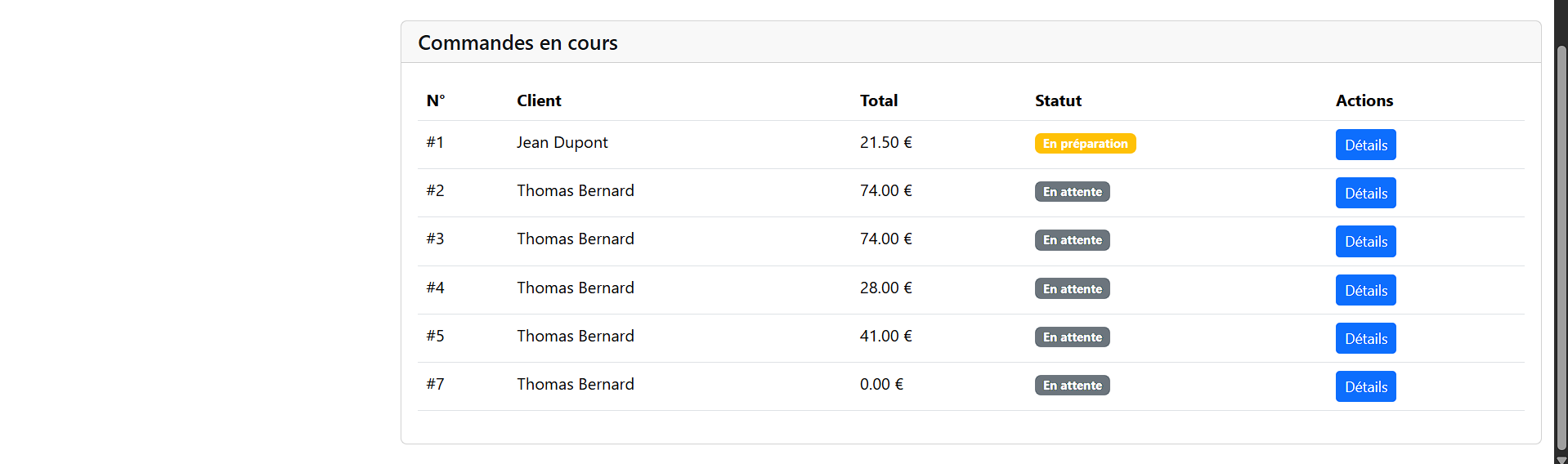
1. Gestion des tables



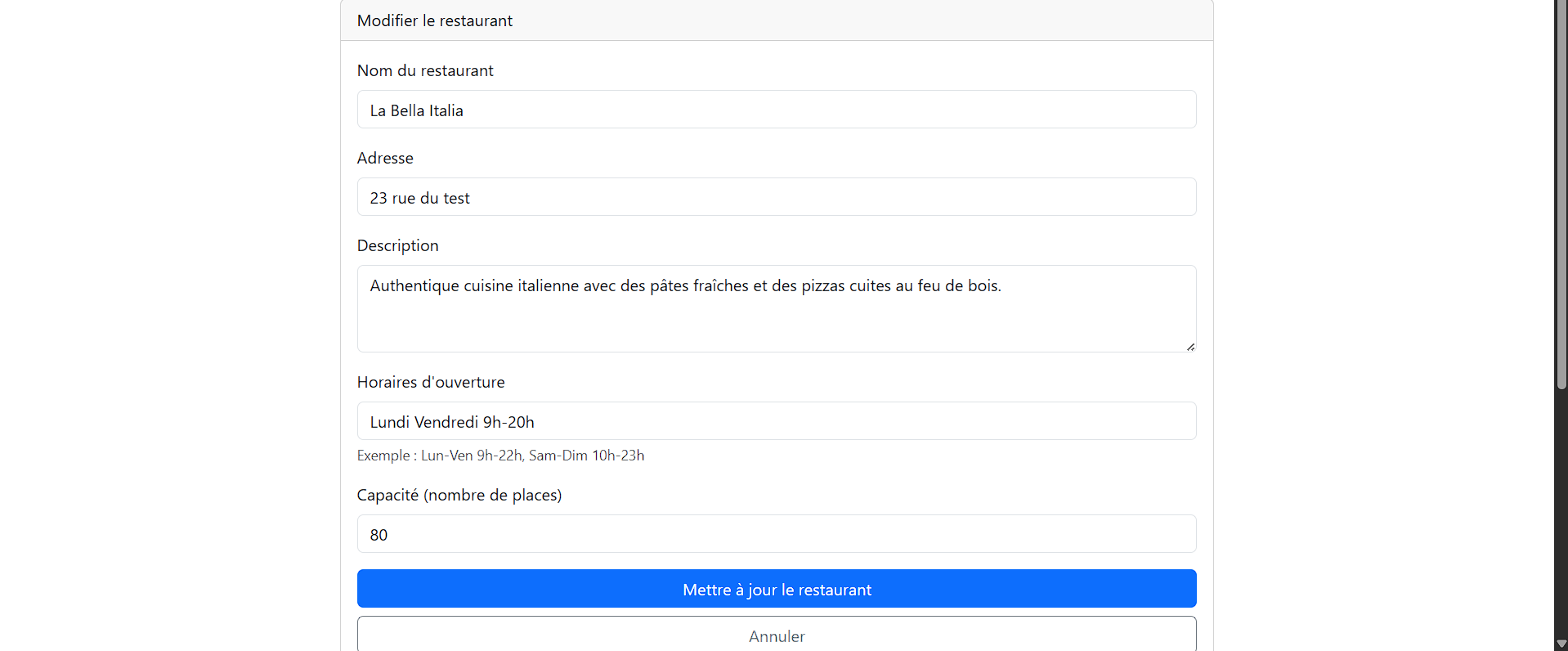
1. Gestion du menu



1. Gestion des commandes

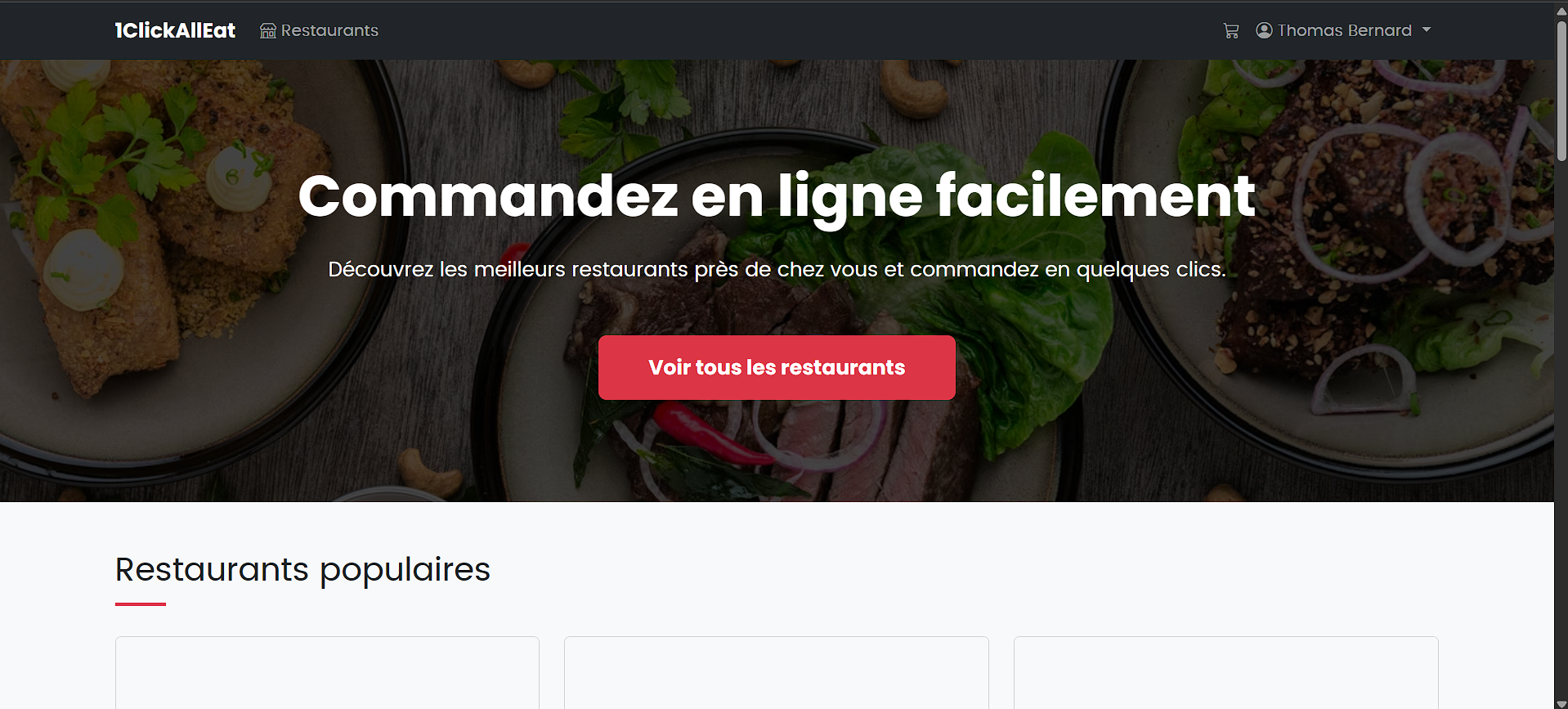


1. Paramètres du restaurant

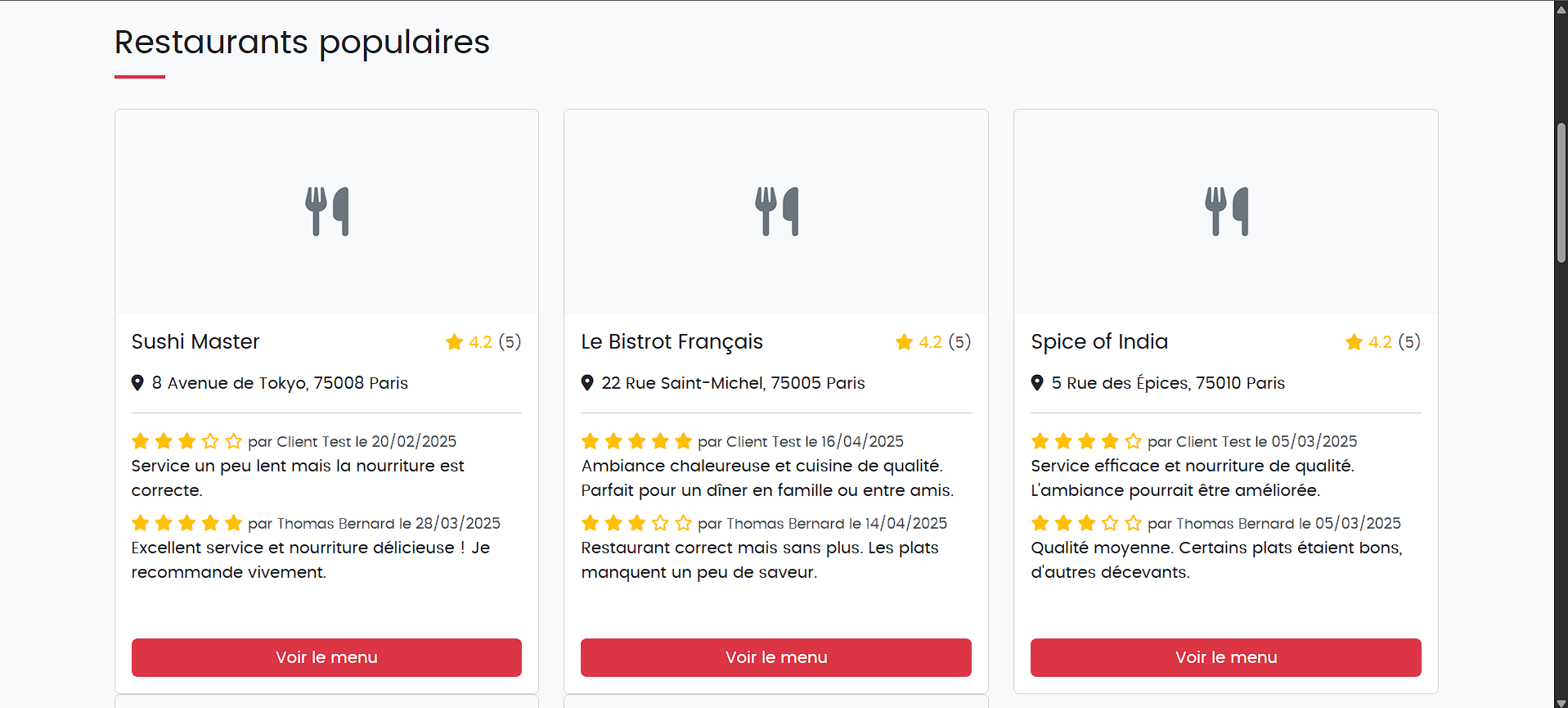


**Section 2: Interface Client**

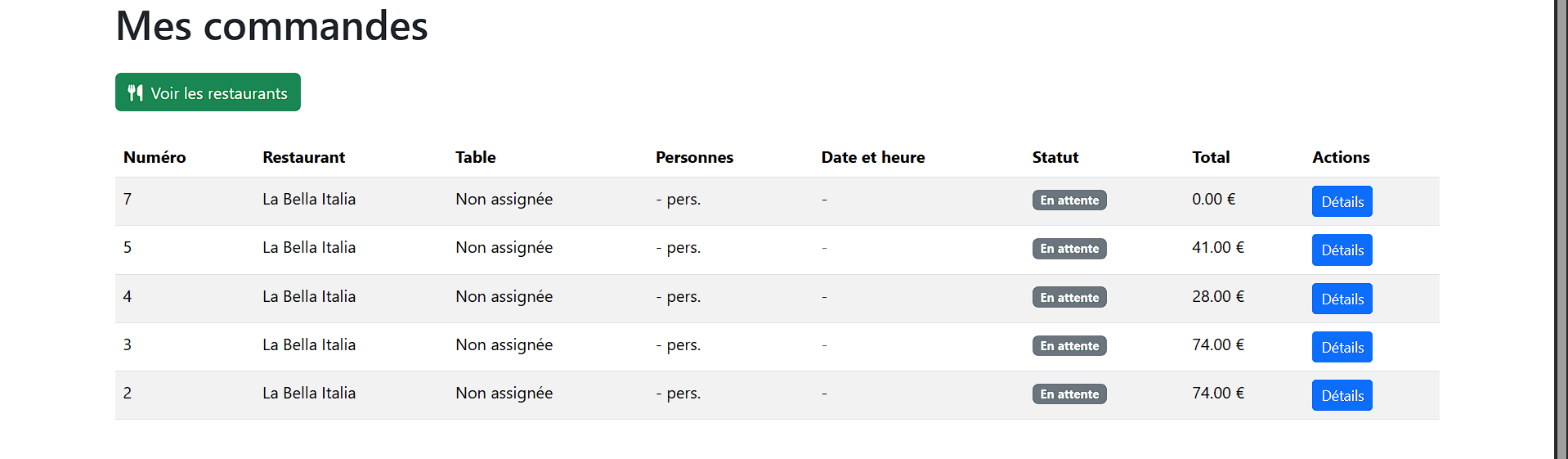
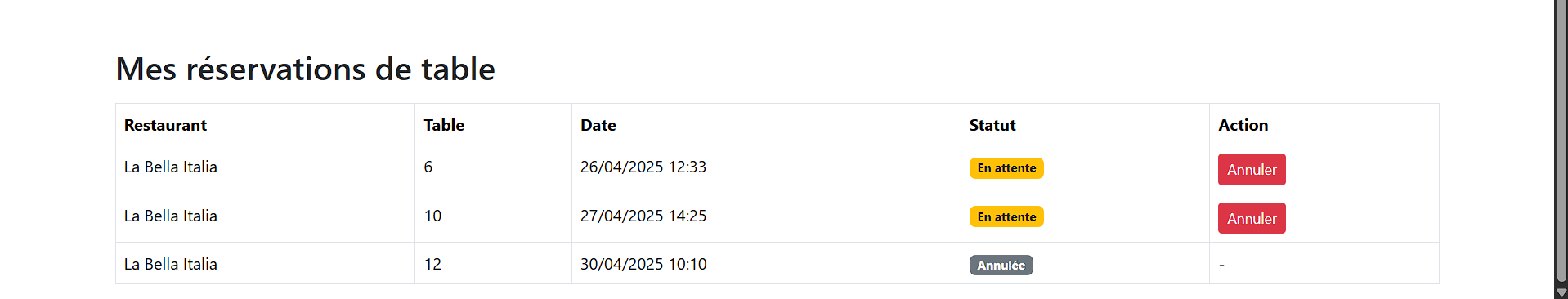
1. Parcourir les restaurants

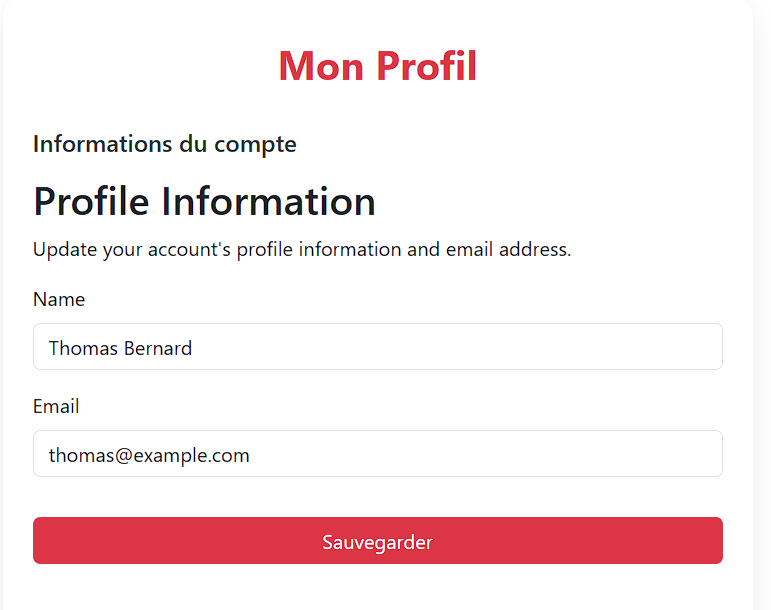


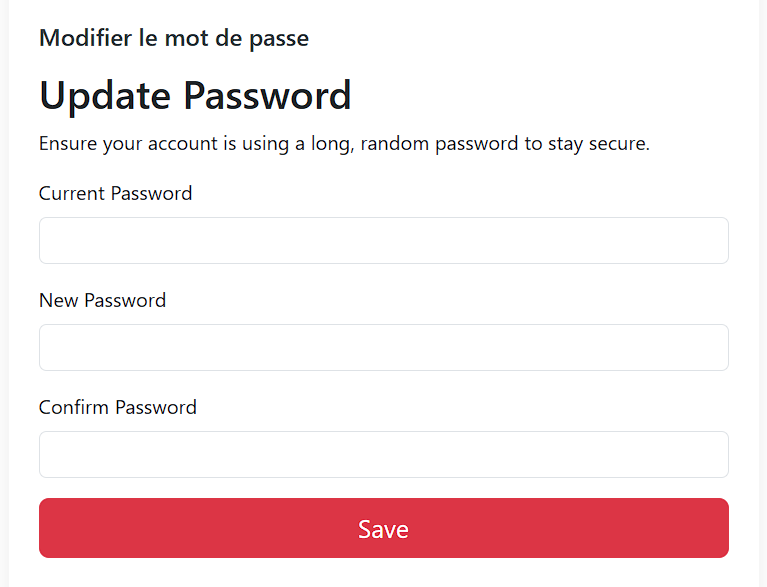
1. Commander en ligne

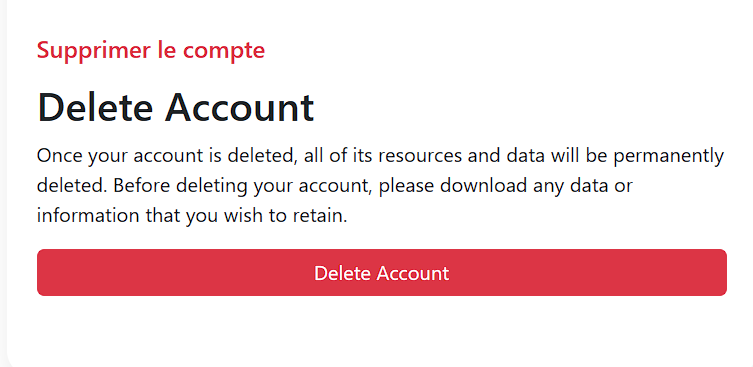


1. Suivi des commandes et réservations



1. Profil utilisateur





**Section 1: Interface Restaurateur**

**1. Dashboard principal**

Le Dashboard Restaurateur est votre centre de contrôle principal pour gérer votre établissement sur la plateforme 1ClickAllEat. Il affiche:

* Le nom de votre restaurant (ex: La Bella Italia)
* Navigation principale: Informations, Menu, Tables
* Statistiques essentielles:
  + Commandes aujourd'hui (nombre)
  + Chiffre d'affaires (CA) du jour
  + Note moyenne et nombre d'avis (ex: 3.8/5 basé sur 5 avis)
  + Commandes en attente (nombre)
* Top 5 des plats les plus vendus avec quantités et CA générés
* Statistiques de performance par période (aujourd'hui, semaine, mois, total)

Le bouton "Modifier le restaurant" en haut à droite vous permet d'accéder rapidement aux paramètres de votre établissement.

**2. Gestion des tables**

L'onglet "Tables" vous permet de visualiser et gérer l'ensemble des tables de votre restaurant:

* Liste complète avec: Numéro, Capacité, Statut (Disponible/Occupée)
* Affichage des réservations en cours pour les tables occupées
* Pour chaque table, vous pouvez:
  + Modifier les informations (icône crayon)
  + Supprimer la table (icône poubelle)
* Bouton "Ajouter une table" pour créer une nouvelle table avec:
  + Numéro de table
  + Capacité (nombre de personnes)

Cette vue vous aide à optimiser l'occupation de votre restaurant et à suivre les réservations en temps réel.

**3. Gestion du menu**

L'onglet "Menu" vous permet de créer et gérer l'ensemble de votre carte:

* Organisation par catégories: Entrées, Plats principaux, Desserts, etc.
* Pour chaque plat, vous visualisez:
  + Nom du plat
  + Description
  + Prix
  + Actions (modifier/supprimer)
* Bouton "Ajouter un plat" pour enrichir votre menu avec:
  + Nom du plat
  + Description détaillée
  + Prix en euros
  + Catégorie (à sélectionner dans le menu déroulant)

Gardez votre menu à jour pour offrir la meilleure expérience à vos clients.

**4. Gestion des commandes**

La section "Commandes en cours" vous permet de suivre toutes les commandes reçues:

* Numéro de commande
* Nom du client
* Montant total
* Statut (En préparation, En attente)
* Action "Détails" pour consulter le détail complet

Cette vue vous aide à traiter efficacement les commandes et à suivre leur progression.

**5. Statistiques et performances**

Le dashboard principal affiche des indicateurs clés de performance:

* Top 5 des plats les plus populaires avec:
  + Nom du plat
  + Quantité vendue
  + Chiffre d'affaires généré
* Récapitulatif financier par période:
  + Aujourd'hui (0 commandes, 0.00 €)
  + Cette semaine (6 commandes, 238.50 €)
  + Ce mois (6 commandes, 238.50 €)
  + Total (6 commandes, 238.50 €)

Ces données vous permettent d'analyser les tendances et d'optimiser votre offre.

**6. Paramètres du restaurant**

Le formulaire "Modifier le restaurant" vous permet de mettre à jour les informations essentielles:

* Nom du restaurant
* Adresse complète
* Description détaillée
* Horaires d'ouverture (format suggéré: Lun-Ven 9h-22h, Sam-Dim 10h-23h)
* Capacité totale (nombre de places)

Gardez ces informations à jour pour que les clients puissent trouver facilement votre établissement.

**Section 2: Interface Client**

**1. Parcourir les restaurants**

La page d'accueil client présente:

* Bannière principale "Commandez en ligne facilement"
* Courte description: "Découvrez les meilleurs restaurants près de chez vous et commandez en quelques clics"
* Bouton "Voir tous les restaurants" pour accéder au catalogue complet
* Section "Restaurants populaires" présentant les établissements les plus appréciés
* Filtres de recherche avec:
  + Nom du restaurant
  + Note minimale
  + Adresse/Ville (Ex: Paris)
  + Tri (A-Z)

Cette interface permet aux clients de découvrir rapidement des restaurants qui correspondent à leurs préférences.

**2. Réserver une table**

La section "Mes réservations de table" permet aux clients de:

* Visualiser leurs réservations existantes avec:
  + Nom du restaurant
  + Numéro de table
  + Date et heure
  + Statut (En attente, Annulée)
  + Action possible (Annuler)
* Réserver de nouvelles tables dans les restaurants participants

Les réservations sont synchronisées avec le dashboard restaurateur, permettant une gestion efficace des places.

**3. Commander en ligne**

Les clients peuvent passer commande directement depuis la plateforme:

* Parcourir le menu du restaurant organisé par catégories
* Ajouter des plats à leur panier
* ²Définir la quantité pour chaque article
* Voir le total de leur commande en temps réel
* Valider et payer en ligne

**4. Suivi des commandes et réservations**

La section "Mes commandes" permet aux clients de:

* Voir l'historique complet de leurs commandes
* Consulter le statut de chaque commande (En attente, En préparation)
* Accéder aux détails complets (articles, prix, restaurant)
* Voir le montant total de chaque commande
* Les commandes sont numérotées pour faciliter le suivi

Cette transparence améliore l'expérience client et réduit les demandes d'information.

**5. Profil utilisateur**

Le menu utilisateur (accessible depuis l'avatar en haut à droite) permet d'accéder à:

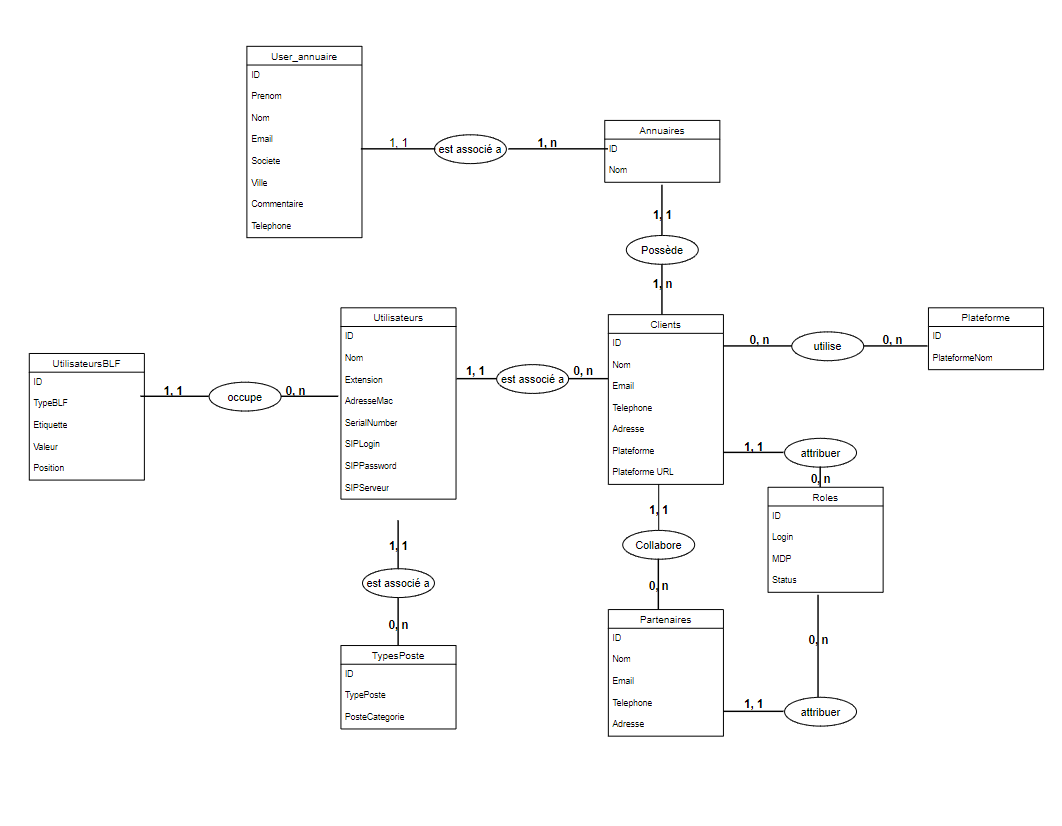
* "Mon profil": informations personnelles
* "Mes commandes": historique des commandes
* "Déconnexion": quitter la session en cours

Les clients peuvent ainsi gérer facilement leur compte et accéder à leur historique.

**Annexe \* : Telora**

**Cahier des charges :**

**Diagramme UML :**

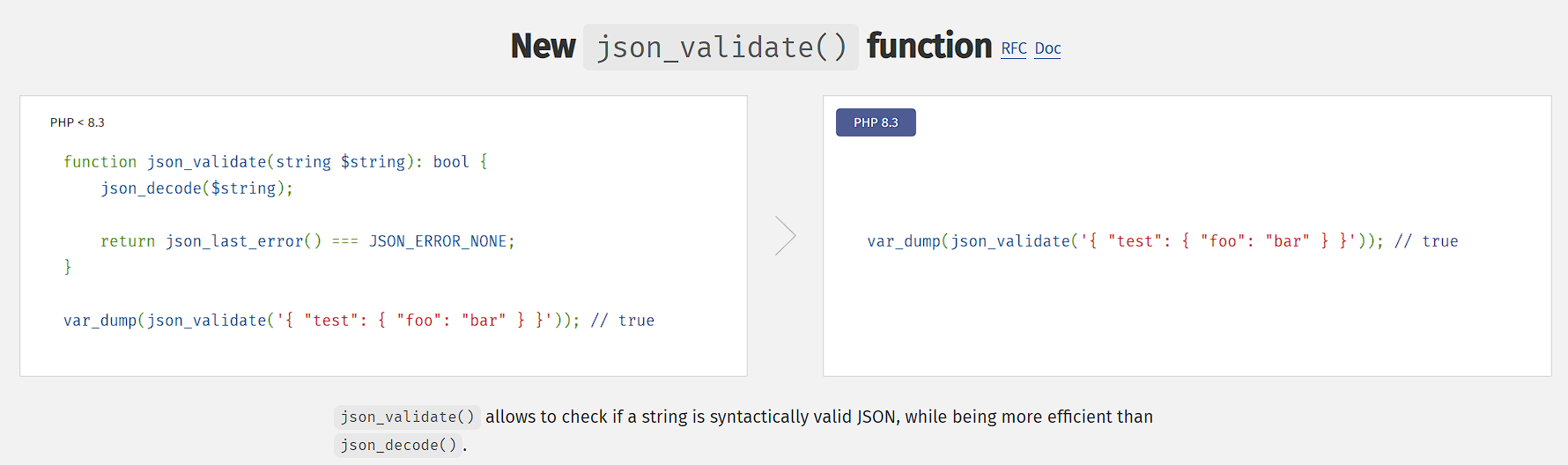
****

**Merise**

Annexe Veille :

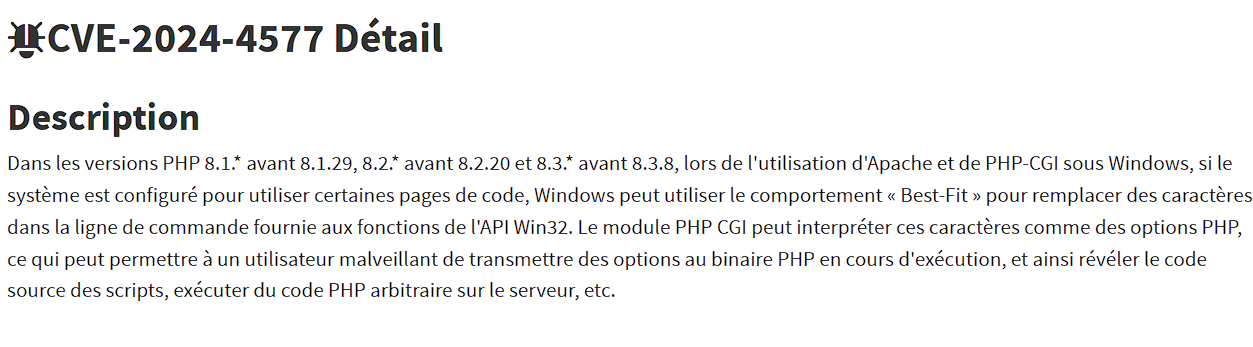
PHP : Nouveautés, Sécurité, Bonnes Pratiques

PHP 8.3 : Nouveautés clés (https://www.php.net/releases/8.3/en.php)

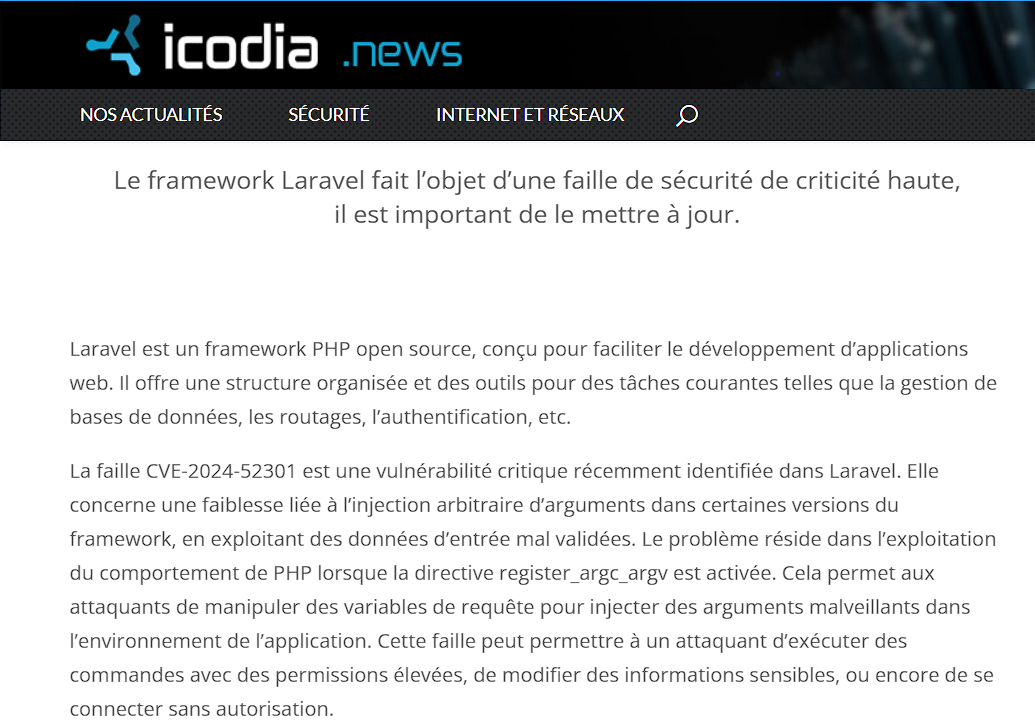
* Constantes typées : PHP 8.3 permet désormais de typer explicitement les constantes de classe, renforçant ainsi la robustesse du typage.
* Clonage profond des propriétés readonly : Amélioration de la gestion des propriétés en lecture seule, facilitant la duplication d'objets complexes.
* Fonction json\_validate() : Nouvelle fonction pour valider la syntaxe JSON sans le décoder, optimisant ainsi les performances.
* Améliorations générales : Optimisations de performance, corrections de bugs et nettoyage du code.

Vulnérabilité CVE-2024-4577 (https://nvd.nist.gov/vuln/detail/cve-2024-4577)

* Description : Faille critique affectant PHP-CGI sur Windows, permettant à un attaquant de passer des options malveillantes à l'exécutable PHP, pouvant conduire à l'exécution de code arbitraire.
* Versions concernées : PHP 8.1 < 8.1.29, PHP 8.2 < 8.2.20, PHP 8.3 < 8.3.8.
* Recommandation : Mettre à jour vers les versions corrigées dès que possible.

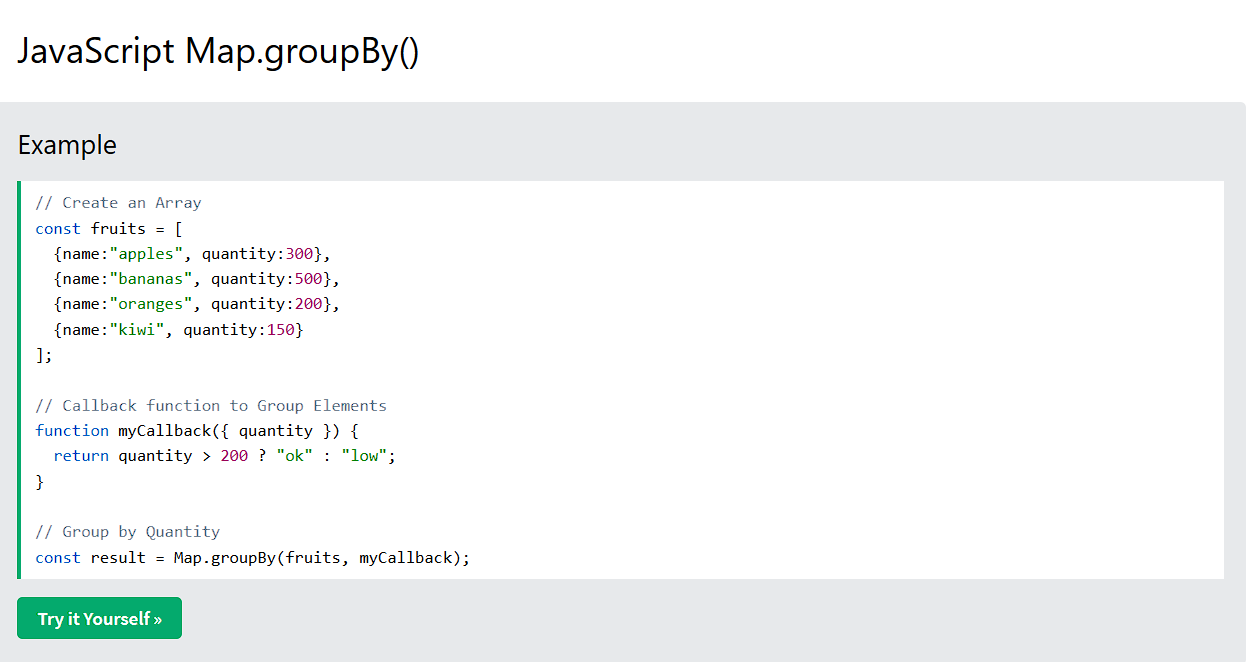


CVE-2024-52301 – Injection d'arguments via register\_argc\_argv (https://news.icodia.com/actualites-icodia/faille-de-securite-sur-le-framework-laravel)

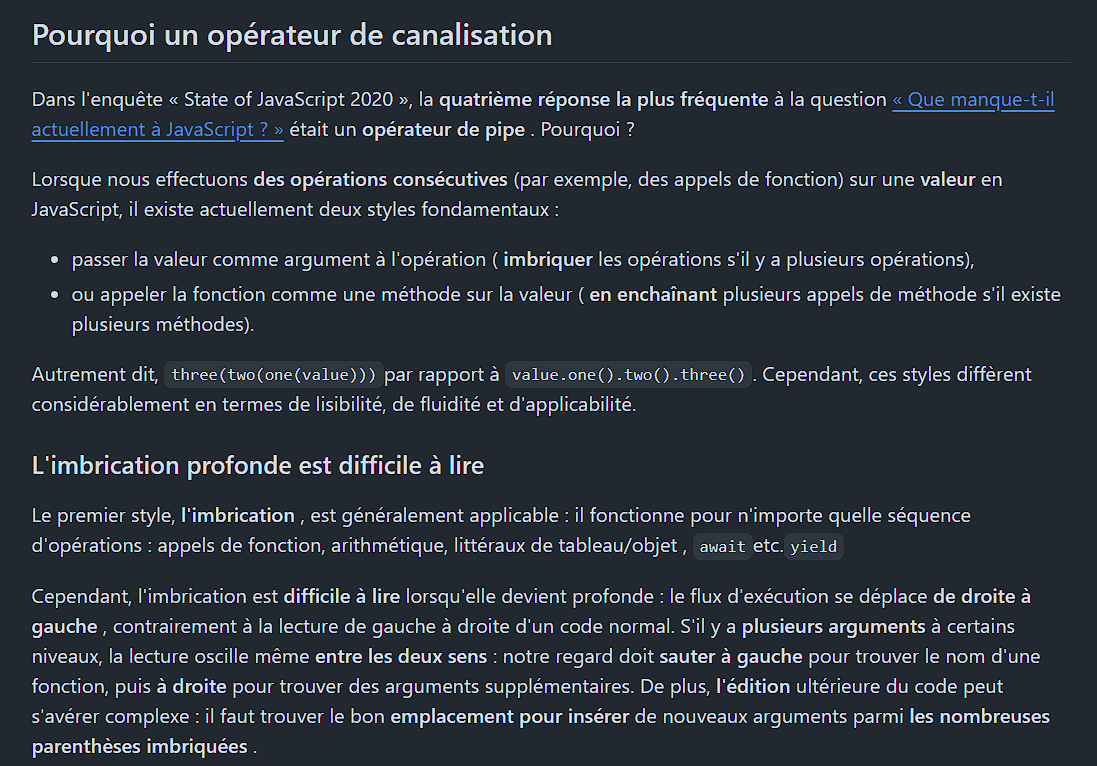
* Date de publication : 21 novembre 2024
* Versions affectées :
  + Laravel 6 < 6.20.45
  + Laravel 7 < 7.30.7
  + Laravel 8 < 8.83.28
  + Laravel 9 < 9.52.17
  + Laravel 10 < 10.48.23
  + Laravel 11 < 11.31.0
* Description : Une vulnérabilité critique permet à un attaquant d'injecter des arguments arbitraires en exploitant la directive PHP register\_argc\_argv. Cela peut conduire à l'exécution de commandes non autorisées ou à l'accès à des informations sensibles.
* Solution : Mettre à jour Laravel vers une version corrigée. Si la mise à jour immédiate n'est pas possible, désactivez register\_argc\_argv dans la configuration PHP, en étant conscient des effets secondaires potentiels.

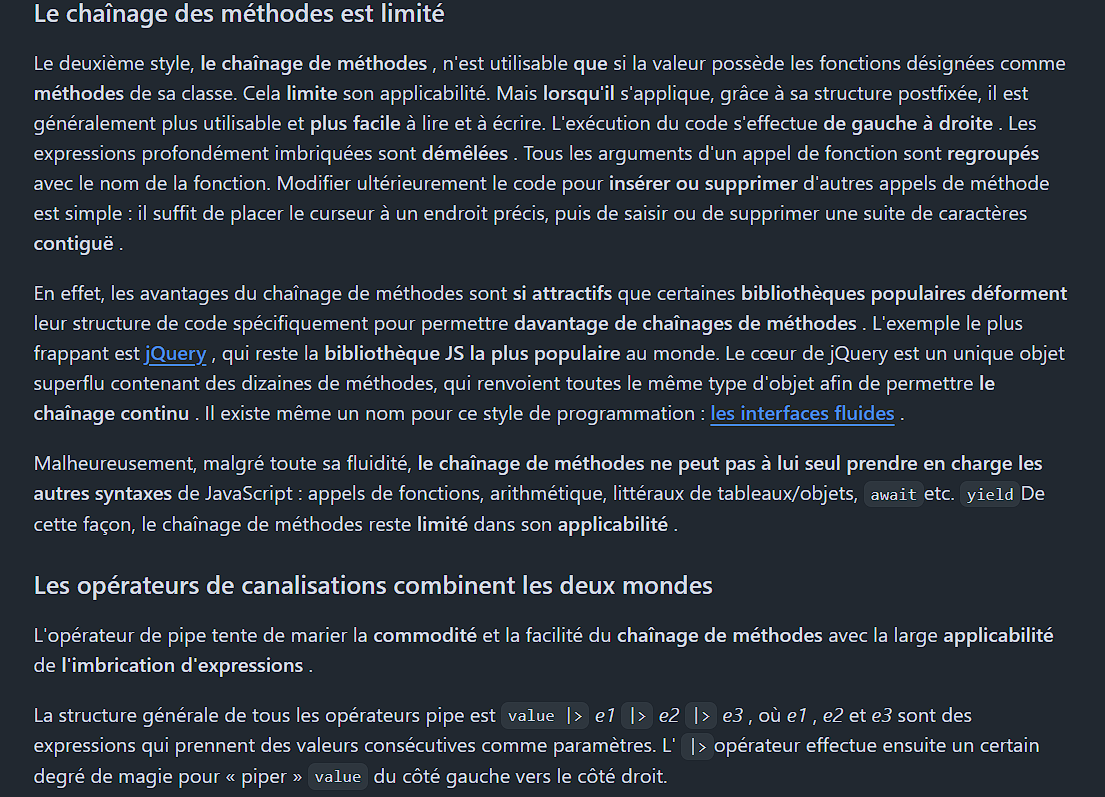
JavaScript : ES2024, Pipeline Operator, Temporal API (https://www.w3schools.com/js/js\_2024.asp)

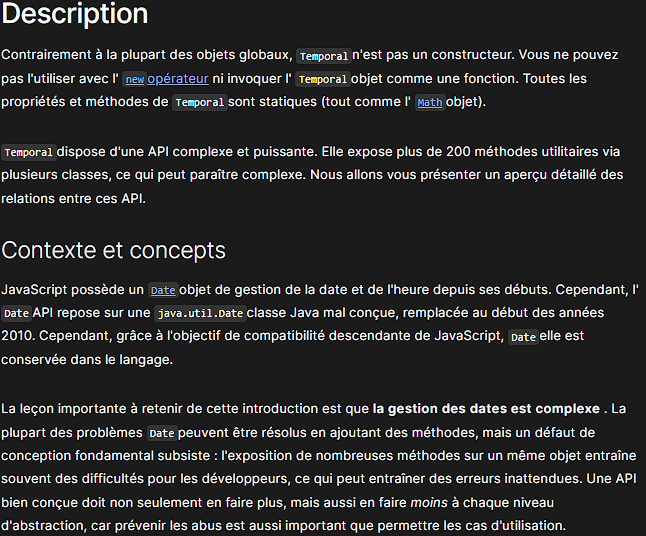
ES2024 : Nouveautés majeures

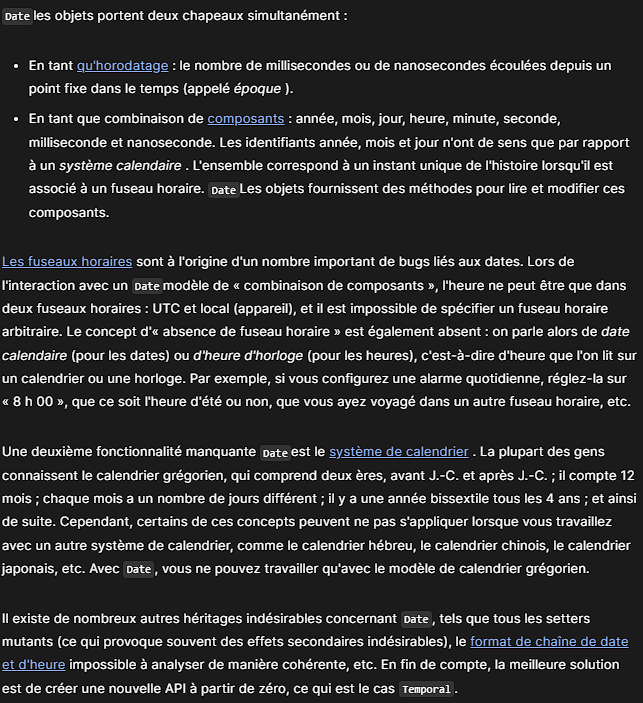
* Object.groupBy() et Map.groupBy() : Nouvelles méthodes pour regrouper les éléments d'un objet ou d'une map selon une fonction de regroupement.
* Temporal API : Introduction de nouvelles classes pour une gestion précise des dates et heures, remplaçant l'objet Date.

Pipeline Operator (|>) (https://github.com/tc39/proposal-pipeline-operator)

* Fonctionnalité : Permet de chaîner les appels de fonctions de manière plus lisible, en passant la sortie d'une fonction comme entrée de la suivante.
* Avantage : Améliore la clarté du code en évitant les imbrications complexes.

Temporal API (http://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\_Objects/Temporal)

* Objectif : Fournir une API moderne pour la gestion des dates et heures, avec prise en charge des fuseaux horaires et des calendriers.
* Avantages : Précision accrue, évite les bugs liés à l'objet Date, et offre plus de flexibilité pour les calculs temporels.



# LDAP – Telora

## Cahier des charges détaillé

**1. Contexte et Présentation générale**

Telora est une application web de gestion de contacts, d’annuaires téléphoniques et d’utilisateurs, pensée pour un fonctionnement en marque blanche. Elle vise à permettre à différents types d’utilisateurs (Admins, Partenaires, Clients finaux) de gérer leurs propres données, via une interface moderne, responsive et sécurisée.

Le projet est développé en PHP/MySQL côté serveur, avec une interface HTML/CSS/JS enrichie par TailwindCSS et FontAwesome.

**2. Objectifs principaux**

- Centraliser la gestion des contacts et des utilisateurs pour des clients variés (partenaires, entreprises, utilisateurs finaux).

- Offrir une séparation stricte des accès selon le rôle.

- Proposer une expérience utilisateur fluide, moderne et sécurisée.

- Permettre l’import/export de contacts au format CSV, avec robustesse face aux erreurs de format.

**3. Public cible**

- Administrateurs système (super-admins)

- Partenaires (entreprises clientes de Telora)

- Clients finaux (utilisateurs des partenaires)

**4. Expression des besoins et fonctionnalités**

**4.1. Gestion multi-niveaux et rôles**

- Admins : gestion globale des partenaires, clients, utilisateurs, et contacts.

- Partenaires : gestion de leurs propres clients finaux et de leurs contacts.

- Clients finaux : gestion de leurs propres contacts et informations.

**4.2. Gestion des entités**

- CRUD complet (création, lecture, modification, suppression) pour :

- Clients

- Utilisateurs

- Contacts de l’annuaire

- Import/export CSV des contacts (encodage UTF-8, format compatible)

- Recherche, filtrage, pagination des contacts

**4.3. Sécurité et navigation**

- Séparation stricte des accès selon le rôle (aucune fuite de données entre clients)

- Gestion du contexte de navigation via la session PHP

- Redirection automatique vers la page de connexion en cas d’accès non autorisé

- Logs d’erreur activés pour le suivi des bugs (sans fuite d’informations sensibles)

- Jamais de modification des variables de session à partir de l’URL côté client

- Vérification stricte des droits d’accès à chaque page sensible

**4.4. Interface utilisateur**

- Interface responsive, moderne, centrée, avec header harmonisé sur toutes les pages

- Utilisation de containers pour centrer et limiter la largeur du contenu (max 1200px)

- Design cohérent entre tous les formulaires et tableaux

- Utilisation de boutons modernes, feedback utilisateur (succès/erreur), modals pour les actions sensibles

- Intégration de TailwindCSS et FontAwesome

**4.5. Points d’attention**

- Maintien du contexte utilisateur lors de la navigation (pas de fuite de session)

- Cohérence visuelle et UX sur toutes les pages (header, boutons, tableaux, formulaires)

- Robustesse de la gestion des imports/exports CSV (gestion des erreurs de format)

- Tests sur différents rôles et scénarios d’accès

**5. Contraintes techniques**

- Technologies : PHP (backend), MySQL (base de données), HTML/CSS/JS (frontend), TailwindCSS, FontAwesome

- Organisation : séparation claire entre la logique métier (PHP, dossier classes/api), les vues (HTML, dossiers admin, annuaire, clientlist, utilisateurdetail…), et le style (CSS, partials)

- Sécurité : sessions PHP, contrôle d’accès, logs d’erreur, protection contre la modification des sessions côté client

- Ergonomie : centrage du contenu, largeur limitée, responsive design

- Tests : activation des logs, rédaction d’un cahier de tests fonctionnels, tests manuels/automatisés selon disponibilité

**6. Architecture technique (modules/dossiers)**

- admin/ : gestion des administrateurs et des partenaires

- annuaire/ : gestion des contacts et de l’annuaire téléphonique

- clientlist/ et clientdetail/ : gestion des clients

- utilisateurdetail/ : gestion des utilisateurs

- api/ : endpoints pour l’interface ou l’intégration externe

- classes/ : logique métier et objets PHP

- config/, database/ : configuration et scripts SQL

- login/ : authentification et gestion des sessions

- partials/ : fragments d’interface (header, footer…)

- utils/ : utilitaires divers

- docs/, Documentation\_LDAP/ : documentation technique et d’intégration

**7. Livrables attendus**

- Code source complet (PHP, CSS, JS, SQL)

- Documentation technique (structure, installation, sécurité)

- Cahier de tests fonctionnels (scénarios d’accès, sécurité, import/export)

- Fichiers d’exemple pour l’import/export CSV

**8. Déploiement et maintenance**

- Déploiement sur un serveur compatible PHP/MySQL, avec configuration sécurisée (HTTPS recommandé)

- Scripts d’installation et de migration de base de données fournis

- Procédures de sauvegarde et de restauration des données à documenter

**9. Sécurité et RGPD**

- Protection des données personnelles (sessions, accès, logs limités)

- Conformité RGPD à assurer (mentions légales, droits des utilisateurs)

- Certificat SSL recommandé pour toutes les communications

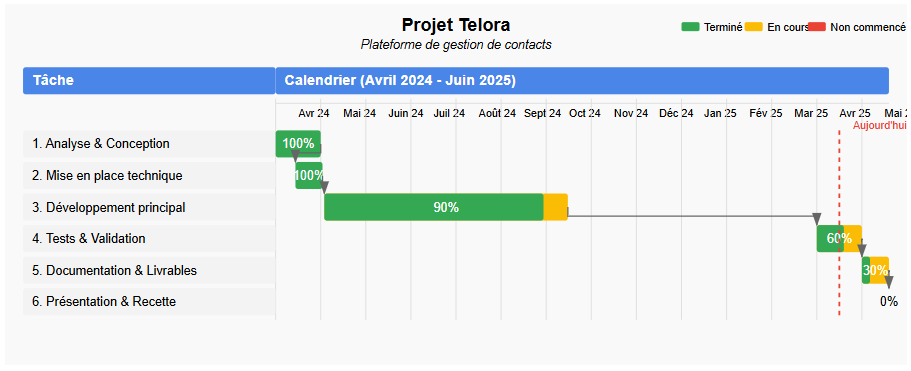
**10. Tests et validation**

- Rédaction d’un cahier de tests couvrant tous les rôles et scénarios critiques

- Tests d’import/export CSV avec gestion des erreurs

- Validation finale par démonstration auprès du tuteur

# Diagramme de gantt

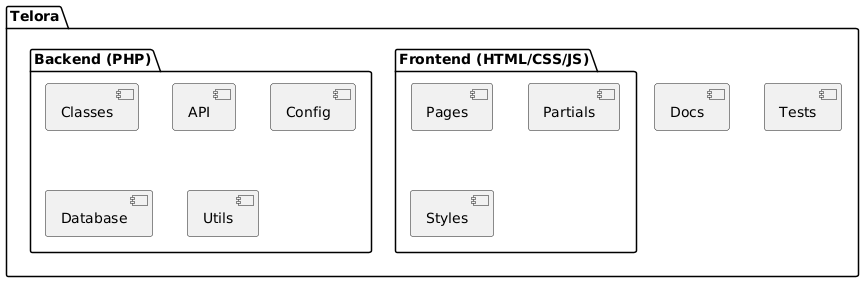


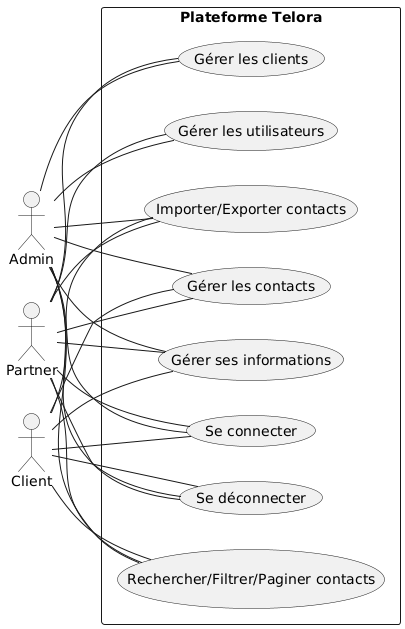
# LDAP - Telora

## UML

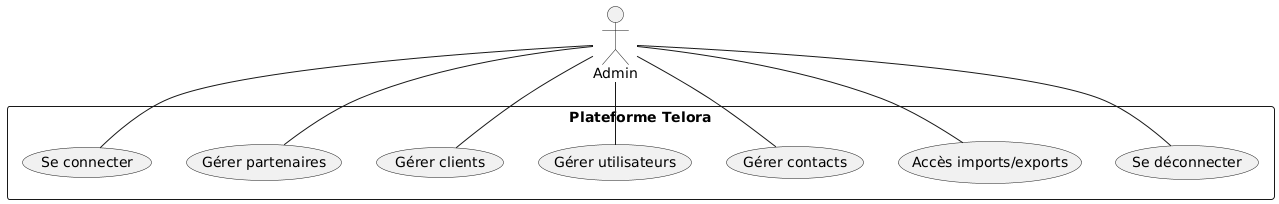
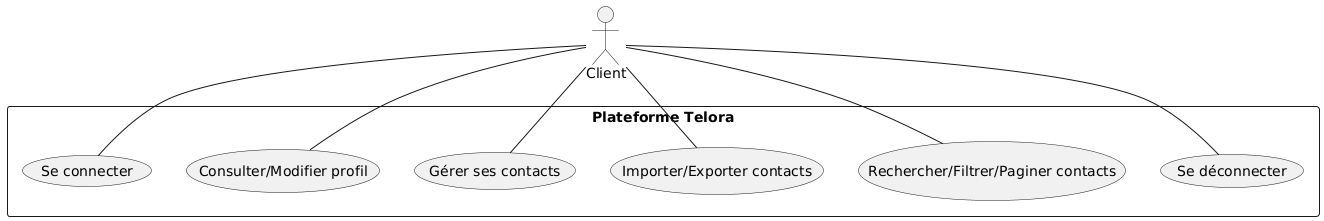
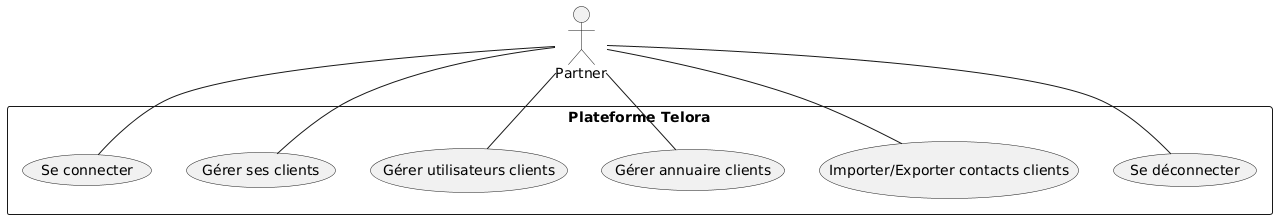
### 1. Diagramme de contexte

### 2. Diagramme de package

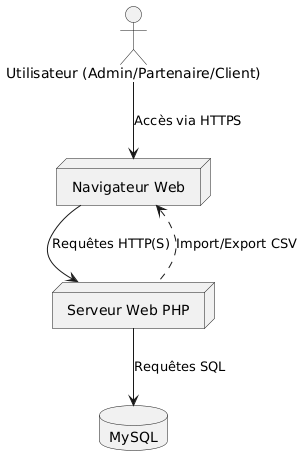




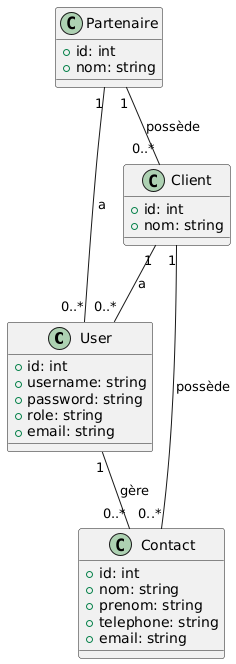
### 3. Diagramme use\_case



### 4. Cartographie technique



### 5. Diagramme de classe



# LDAP – Telora

## Merise – Modélisation BDD

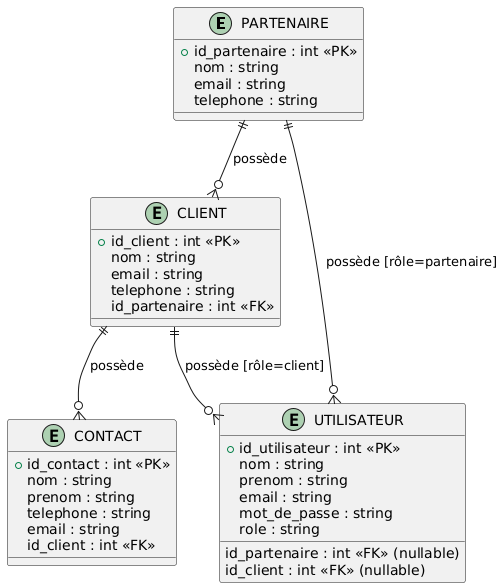
### 1. Dictionnaire de données

### 2. MCD (Modelé conceptuel de données)

### 

### 

### 3. MLD (Modèle logique de données)



### 4. MPD (Modèle physique de données)

# 1ClickAllEat

## Extraits de code commentés

### 1. Authentification et gestion des rôles

# 

# Explication détaillée :

# session\_start()

# démarre la session PHP, permettant d’accéder aux variables de session pour suivre l’état de connexion de l’utilisateur.

# La condition

# if (!isset($\_SESSION['role']))

# vérifie si l’utilisateur est authentifié (un rôle doit être défini en session après connexion).

# Si l’utilisateur n’est pas connecté, il est redirigé vers la page de login, ce qui protège toutes les pages sensibles contre l’accès non autorisé.

# Les variables de session comme

# role

# et

# partner\_id

# sont utilisées pour adapter dynamiquement l’interface et les droits d’accès. Par exemple, un partenaire ne pourra voir que ses propres clients, un client que ses propres contacts, etc.

# Ce mécanisme centralise la gestion des droits et garantit la séparation stricte des accès selon le rôle, comme exigé dans le cahier des charges.

### 2. Récupération et affichage paginé des contacts

# Explication détaillée :

# Cette séquence gère la pagination pour l’annuaire des contacts d’un client.

# Elle récupère la page courante et le nombre d’éléments à afficher par page depuis l’URL (GET), avec des valeurs par défaut sécurisées.

# countAnnuaireByClient($clientsId) compte le nombre total de contacts pour le client, ce qui permet de calculer le nombre total de pages.

# L’offset (

# $offset

# ) détermine à partir de quel contact commencer l’affichage pour la page courante.

# getAnnuaireByClientPaginated($clientsId, $offset, $perPage) récupère uniquement les contacts à afficher pour cette page, optimisant les performances même pour de gros volumes de données.

# Ce mécanisme améliore l’ergonomie et la rapidité de navigation dans l’annuaire.

### 3. Modification d’un contact

# Explication détaillée :

# Cette méthode met à jour toutes les informations d’un contact dans la base de données.

# Elle utilise une requête SQL préparée, ce qui protège contre les injections SQL et garantit la sécurité des données.

# Chaque champ du contact (prénom, nom, email, etc.) est lié à un paramètre de la requête pour éviter toute altération malveillante.

# En cas d’erreur lors de la mise à jour (par exemple, si la base est inaccessible), une exception claire est levée, facilitant le debug et la robustesse de l’application.

# Cette méthode est appelée lors de la modification d’un contact depuis l’interface utilisateur.

### 4. Import de contacts depuis un fichier CSV

# Explication détaillée :

# Cette fonction permet d’importer en masse des contacts pour un client à partir d’un fichier CSV.

# Elle lit chaque ligne du fichier, nettoie les données (suppression des caractères spéciaux, normalisation des champs) et insère chaque contact dans la base.

# Elle gère les erreurs de format (colonnes manquantes, emails invalides, etc.) et évite les doublons, assurant la qualité des données importées.

# Ce mécanisme facilite la migration de données ou l’intégration de nouveaux clients disposant déjà d’un annuaire.

### 5. Suppression d’un client

# Explication détaillée :

# Cette méthode supprime un client de la base de données à partir de son identifiant.

# L’utilisation d’une requête préparée garantit la sécurité de l’opération.

# Elle permet de gérer la suppression d’un client depuis l’interface d’administration ou de gestion partenaire, en respectant les droits d’accès.

### 6. Connexion utilisateur

# Explication détaillée :

# Cette méthode gère la connexion d’un utilisateur à l’application.

# Elle recherche l’utilisateur dans la base à partir du login, puis vérifie le mot de passe avec

# password\_verify

# , qui prend en charge le hachage sécurisé.

# Si la connexion réussit, elle initialise la session et stocke l’identifiant et le rôle de l’utilisateur, permettant de personnaliser l’accès à l’application.

# La fonction

# redirectBasedOnRole

# permet de rediriger l’utilisateur vers l’espace qui lui correspond (admin, partenaire, client).

# En cas d’échec, la méthode retourne

# false

# ce qui permet d’afficher un message d’erreur à l’utilisateur.

# LDAP – Telora

## Plan de test qualité

### 1. Objectif

Valider que l’application actuelle répond aux besoins métiers (gestion utilisateurs, clients, partenaires, annuaire), assure la sécurité des données, la robustesse des traitements, et offre une expérience utilisateur cohérente sur tous supports.

### 2. Périmètre du test

**Fonctionnalités principales à tester**

* Authentification (login, logout, inscription)
* Gestion des utilisateurs (création, modification, suppression, affichage détail)
* Gestion des clients (CRUD, affichage liste/détail)
* Gestion des partenaires/restaurants (CRUD, accès admin)
* Gestion de l’annuaire (CRUD contacts, import/export CSV, recherche)
* Gestion des accès différenciés (admin, client)
* Sécurité (chiffrement, gestion des mots de passe)
* Responsive design (desktop, mobile)
* Import/export de données (CSV, LDAP)
* Scripts techniques (test TCP, import LDAP)

**Contraintes techniques**

* PHP custom, pas de framework
* Sécurité basique (chiffrement, gestion sessions)
* Tests techniques manuels/automatisés via scripts dédiés (pas de PHPUnit)
* Pas de CI/CD détecté

### 3. Méthodologie

* Tests manuels sur chaque fonctionnalité métier (CRUD, login, etc.)
* Tests d’intégration sur les interactions modules (ex : ajout client → visible dans annuaire)
* Tests de sécurité (mot de passe, session, accès différencié)
* Tests de performance sur les scripts d’import/export et requêtes lourdes
* Tests de compatibilité (navigateurs, responsive)
* Tests techniques via scripts dédiés (testTCP, LDAP\_test)

### 4. Tableau du plan de test

### 5. Exécution, reporting et validation

* Toutes les anomalies sont tracées dans un outil de suivi (ex : GitHub Issues).
* Un rapport de test est généré à chaque fin de cycle.
* La mise en production n’est validée que si tous les tests critiques sont passés et les anomalies bloquantes corrigées.

### 6. Conclusion

Ce plan de test qualité est adapté à la structure et aux fonctionnalités réelles de ton application. Il doit être révisé à chaque évolution majeure ou correction.