Rapport

INFO3 – Base de données NoSQL

Gestion de Restaurant

Groupe : F

Auteurs : OUBERKA Mohamed

BOUCHELLAL Imad-Eddine

OUKACI Djedjiga

ADRAR Céline

Encadré par : Eddy Kiomba

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Document Version : 1.0

Submission date : 21-11-2025

Contents

1 Contexte et problématique

1.1 Contexte général

Le projet s'inscrit dans le domaine de la gestion numérique des commandes pour la restauration. L'objectif principal est d'automatiser l'ensemble du flux opérationnel : prise de commande, transmission en cuisine, service en salle et facturation.

La solution technique repose sur une base de données MongoDB qui offre plusieurs avantages structurels : Modélisation flexible : stockage sous forme de documents permettant d'imbriquer les détails des commandes. Gestion des relations : liens logiques entre serveurs, commandes, tables et plats. Capacités d'analyse : utilisation des pipelines d'agrégation pour générer des statistiques précises (ventes, performances, popularité des produits).

Ce type d'architecture répond aux exigences des restaurants modernes souhaitant fluidifier leur service et minimiser les erreurs humaines grâce à un suivi en temps réel.

1.2 Problématique

La gestion traditionnelle sur support papier présente des limites opérationnelles majeures : erreurs de transmission vers la cuisine, délais de mise à jour des statuts, manque de visibilité globale sur l'activité et difficulté à établir des statistiques fiables.

L'application développée répond à ces enjeux à travers cinq axes fonctionnels : Dématérialisation de la prise de commande. Suivi du cycle de vie des commandes en temps réel (de la création au paiement). Optimisation de la communication salle-cuisine. Analyse de la performance via des indicateurs statistiques.

1.3 Acteurs du système

Le système distingue quatre profils d'utilisateurs, définis par leur rôle : Serveur : Saisit les commandes (Type et quantités), valide le service des plats et l'encaissement de la commande. Cuisinier : Consulte la file d'attente des commandes et met à jour leur avancement (en préparation, prêt). Manager / Administrateur : Administre le catalogue des produits (prix, stock), gère les comptes utilisateurs et analyse les tableaux de bord statistiques.

1.4 Model de données

Les données sont structurées autour de quatre collections principales : Collection Users : Centralise les comptes du personnel avec leurs rôles respectifs. Collection Plats : Référence le catalogue produits (nom, catégorie, prix). Collection Orders : Constitue le cœur du système. Elle lie une table et un serveur à une liste de produits, tout en historisant le statut, le montant total et l'horodatage.

1.5 Relations fonctionnelles

L'architecture des données respecte les règles de gestion suivantes : Une commande est unique à une table donnée (relation 1-1 à un instant T). Un serveur gère de multiples commandes (relation 1-N). Une commande est composée de plusieurs plats (relation N-N matérialisée par des sous-documents).

2 Planification du projet

Cette section détaille la charge de travail prévisionnelle pour une équipe de quatre développeurs, couvrant l'analyse, le développement (backend/frontend) et la livraison.

Analyse, conception et documentation (4h) : Définition des besoins et modélisation de la base de données. Développement Backend (8h) : Architecture de l'API, implémentation de la logique métier des commandes, gestion du catalogue, pipelines d'agrégation et sécurisation des accès, assurés principalement par deux développeurs. Développement Frontend (4h) : Conception et intégration des interfaces utilisateurs pour les serveurs, la cuisine, l'administration et le tableau de bord statistique, menées par un développeur orienté UI/UX. Assurance qualité et tests (2h) : Mise en place des jeux d'essai, exécution des tests fonctionnels et techniques, correction des anomalies, pris en charge par un membre référent qualité. Livraison et support (6h) : Préparation du livrable final, rédaction du rapport technique, préparation de la soutenance et support lors de la démonstration, répartis entre l'ensemble des quatre membres.

3 Périmètre fonctionnel

3.1 Fonctionnalités standard

Le socle de l'application comprend les modules suivants : Administration du catalogue : Création, modification et suspension des plats. Module de commande : Interface de prise de commande avec gestion des quantités. Affichage cuisine : File d'attente interactive permettant aux cuisiniers de changer le statut des plats. Encaissement : Clôture de la commande et enregistrement de la transaction. Historique : Journal complet des commandes avec filtres de recherche. Gestion des utilisateurs : Création et suppression (serveurs, cuisiniers, administrateurs) avec gestion des rôles et droits d'accès par profil.

3.2 Fonctionnalités avancées (Agrégations MongoDB)

Cinq modules statistiques exploitent la puissance des agrégations : Chiffre d'affaires quotidien : Calcul des recettes journalières par regroupement temporel, avec visualisation en histogramme. Statistiques par catégories : Répartition du chiffre d'affaires sur les catégories de produits (Entrée, Plat, Dessert, Boisson) via un diagramme circulaire et classement. Palmarès des ventes : Identification des produits les plus performants par déconstruction des lignes de commande et classement des best-sellers. Performance individuelle : Suivi du volume de ventes en chiffre d'affaires et nombre de commandes par serveur, avec indicateurs de performance. Indicateurs de synthèse : Tableau de bord global affichant : Chiffre d'affaires total Temps moyen de service Meilleur serveur Plat le plus vendu.

4 Proposition de schéma de données orienté application

4.1 Structure des Collections

Conformément aux principes NoSQL, notre modélisation privilégie l'accès rapide aux données critiques (lecture des commandes en temps réel) tout en garantissant l'intégrité des références pour les entités de gestion (utilisateurs, commandes).

Voici un schéma logique NoSQL (orienté documents) cohérent avec ce que tu as sur le tableau, sous forme de diagramme textuel.

Collection users user id name password role (server, cook, admin, cashier)

Collection plats plat id name category (Entrée, Plat, Dessert, Boisson) price

Collection orders order id tableId → référence vers tables.id serverId → référence vers users.id createdAt (date-heure) status (pending, in-progress, ready, served, paid) items : tableau d'objets platId → référence vers plats.id quantity unitPrice (dupliqué pour historiser le prix) total

Ce schéma correspond à un modèle "document + références" typique pour MongoDB, où : orders embarque les lignes de commande dans items. Les liens vers tables, users et plats sont gérés par des identifiants, ce qui reste simple à implémenter et à faire évoluer.

4.2 Justification des choix de modélisation

Notre schéma repose sur deux stratégies distinctes selon la nature des données :

A. Utilisation de Références Nous utilisons des références (IDs) pour lier les Orders aux Users. Les entités Serveurs et Commandes ont des cycles de vie indépendants. Un changement de nom de serveur ou la modification d'une information liée à un serveur doit être appliqué une seule fois, sans nécessiter la mise à jour de l'ensemble des commandes historiques qui lui sont associées. Impact : Cela nécessite une jointure applicative (ou un $lookup en agrégation) pour afficher le nom du serveur sur une facture, mais cela garantit une meilleure maintenabilité des données administratives.

B. Utilisation de l'imbrication avec le motif "Snapshot" Les plats commandés (items) sont imbriqués directement dans le document Order. Performance (Localité des données) : Lorsqu'on affiche une commande en cuisine ou sur l'écran du serveur, on a besoin de la liste des plats immédiatement. L'imbrication évite de faire de multiples requêtes pour récupérer les détails de chaque ligne de commande. Historisation (Pattern Snapshot) : Nous copions les champs critiques (name, price) du plat dans la commande au moment de sa création. Pourquoi ? Si le prix du "Burger" change dans le catalogue Plats le mois prochain, l'historique des commandes passées ne doit pas changer. La commande doit refléter le prix au moment de la vente, pas le prix actuel du catalogue.

5 Exemple de Document MongoDB

Pour illustrer concrètement le fonctionnement du schéma de données retenu, cette section présente un exemple réel de document `Order` généré lors des tests de l'application. Cet exemple met en évidence l'utilisation conjointe des références (pour les entités administratives) et de l'imbrication de documents (pour les lignes de commande), conformément aux bonnes pratiques de modélisation NoSQL.

5.1 Document `Order` issu de l'implémentation

Listing 1 – Exemple réel de document Order inséré dans MongoDB


```
{
  "_id": "692ca299ae8566458c4e9987",
  "numeroTable": 11,
  "serveur": "6928e8e8af11df00dce7026b",
  "items": [
    {
      "plat": "6929c06d57edadbba1e06a20",
      "nom": "Ouefs␣Mimosa",
      "quantite": 1,
      "prixUnitaire": 3
    },
    {
      "plat": "6929c07c57edadbba1e06a23",
      "nom": "Coka␣Cola",
      "quantite": 1,
      "prixUnitaire": 3.5
    },
    {
      "plat": "6929c08957edadbba1e06a26",
      "nom": "tagliatelle",
      "quantite": 10,
      "prixUnitaire": 25
    }
  ],
  "statut": "Pay e",
  "dateCreation": "2025-11-30T20:01:29.846+00:00",
  "montantTotal": 256.5,
  "createdAt": "2025-11-30T20:01:29.853+00:00",
  "updatedAt": "2025-11-30T20:01:57.743+00:00",
  "dateService": "2025-11-30T20:01:56.196+00:00",
  "datePaiement": "2025-11-30T20:01:57.743+00:00"
}
```