Cahier des charges: Projet *resto-reservation*

1. Présentation générale

1.1 Contexte

Le projet **resto-reservation** est une application de réservation pour un restaurant fictif, composée de :

- une **application mobile** en Flutter (Android / iOS),
- une **API REST** en Node.js + Express,
- une **base de données** MySQL,
- une **documentation complète** et une **vidéo de démonstration**.

Arborescence générale du dépôt :

```
resto-reservation/
├── frontend/      # Application Flutter
├── backend/       # API Node.js
├── database/      # Scripts et schéma BDD
├── docs/          # Documentation projet
├── demo/          # Vidéo de démonstration
└── README.md      # Documentation principale
```

1.2 Objectifs

- Offrir aux **clients** :
  - la consultation du **menu** du restaurant ;
  - la **réservation de table** sur des créneaux disponibles ;

    - la consultation, la modification et l’**annulation** de leurs réservations.
- Offrir au **personnel (hôte/serveur)** :
    - la consultation des **réservations** par date ;
    - la **validation / le refus** de réservations ;
    - la visualisation de la **capacité par créneau**.
- Offrir une base technique **clairement structurée** avec séparation :
    - frontend,
    - backend,
    - base de données,
    - documentation et tests.

1.3 Identité du restaurant fictif

Ce bloc peut être adapté en fonction de ton concept.

- **Nom** : *Les AL*
- **Type de cuisine** : bistronomique moderne, inspirations du monde
- **Capacité** : 40 couverts (tables de 2 à 6 personnes)
- **Clientèle cible** : jeunes actifs, couples, petits groupes, touristes

---

2. Périmètre fonctionnel

2.1 Acteurs

- **Client**
- **Hôte / Serveur**
- **Administrateur**

2.2 Fonctionnalités – Client (application Flutter)

2.2.1 Accueil

- Affichage d’un écran d’accueil :
  - présentation succincte du restaurant,
  - boutons d’accès rapide :
    - *Voir le menu*
    - *Réserver une table*
    - *Mes réservations*

2.2.2 Gestion du compte

- Inscription :
  - nom
  - e-mail
  - mot de passe
- Connexion / déconnexion
- Stockage local du **token JWT** pour les appels API

2.2.3 Menu du restaurant

- Affichage du menu récupéré depuis l’API :
  - catégories : entrées, plats, desserts, boissons
  - pour chaque plat : nom, description, prix

2.2.4 Création d’une réservation

- Sélection de :
  - la **date**
  - le **créneau horaire** (ex. 12h00, 12h30, 13h00…)
  - le **nombre de personnes**
- Saisie éventuellement du **numéro de téléphone**
- Envoi au backend :
  - vérification de la **disponibilité du créneau**

- création de la réservation si capacité suffisante

2.2.5 Gestion des réservations (client)

- Écran **Mes réservations** :
  - liste des réservations de l'utilisateur
  - indication du **statut** : en attente, confirmée, refusée, annulée
- Possibilités :
  - modifier une réservation (si statut compatible)
  - annuler une réservation

2.2.6 Notifications (bonus)

- Notification ou alerte lors :
  - de la confirmation / du refus
  - du rappel avant l'horaire de réservation

2.3 Fonctionnalités – Hôte / Serveur

2.3.1 Connexion hôte

- Authentification avec un compte de rôle `host` ou `admin`

2.3.2 Vue des réservations

- Liste des réservations :
  - filtrée par **date** (par défaut : date du jour)
  - filtrable par **statut** : en attente, confirmée, refusée, annulée

2.3.3 Gestion des réservations

- Actions sur une réservation :
  - `pending` → `confirmed`
  - `pending` → `rejected`

    - `confirmed` → `cancelled`
- Consultation des détails :
    - client, téléphone, nombre de couverts, date, heure

2.4 Fonctionnalités – Administrateur

- Gestion du **menu** :
    - ajout / modification / suppression de plats
- Gestion des **créneaux et capacité** :
    - configuration des horaires
    - définition de la capacité max par créneau
- Gestion des **utilisateurs** :
    - création de comptes hôtes / admins
    - changement de rôle

2.5 Hors périmètre

- Paiement en ligne (CB, PayPal, etc.)
- Programme de fidélité complexe
- Multi-restaurants

---

3. Règles de gestion

3.1 Gestion des réservations

- Une réservation contient au minimum :
    - utilisateur (client)
    - date
    - créneau horaire
    - nombre de couverts (≥ 1)

    - statut
- Capacité max par créneau : **40 couverts** (paramétrable).
- Lors de la création :
    - le backend calcule le total de couverts déjà réservés pour `date + créneau` sur les statuts `pending` et `confirmed`
    - si `total + nouveaux_couverts` > capacité max → refus (créneau complet)
- **Statuts** :
    - `pending` : en attente
    - `confirmed` : confirmée
    - `rejected` : refusée
    - `cancelled` : annulée

3.2 Authentification & rôles

- Authentification par **e-mail + mot de passe**
- Mots de passe **hachés**
- Utilisation de **JWT** :
    - contient : `userId`, `email`, `role`
- Contrôle d'accès :
    - `client` :
        - accès à ses propres réservations uniquement
    - `host` :
        - accès aux réservations de tous les clients
        - peut changer les statuts
    - `admin` :
        - accès aux fonctionnalités avancées (menu, utilisateurs, etc.)

---

4. Exigences non fonctionnelles

4.1 Performance

- Temps de réponse API : < 1 s en environnement local
- Fluidité satisfaisante sur émulateur Android

4.2 Sécurité

- Hachage des mots de passe (ex. bcrypt)
- Protection des routes sensibles par middleware JWT
- Validation des données côté frontend et backend

4.3 Qualité & maintenabilité

- Code structuré par modules :

  Frontend (`frontend/`)

  - `lib/main.dart` : logique actuelle de l'appli
  - `android/` et `ios/` : configurations plateformes
  - `test/` : tests unitaires Flutter
  - `pubspec.yaml` : gestion des dépendances

  Backend (`backend/`)

  - `server.js` : serveur Express + routes
  - `package.json` / `package-lock.json` : dépendances
  - `.env` / `.env.example` : configuration
  - `README.md` : guide d'installation backend

  Base de données (`database/`)

  - `init.sql` : création et peuplement initial
  - `schema.png` : schéma relationnel
  - `README.md` : instructions d'import / config

  Documentation (`docs/`)

  - `CAHIER_DES_CHARGES.pdf`

- `api/endpoints.md`
- `guide/gestion-taches.md`
- `guide/getting-started.md`
- `roles.md`
- `DOCUMENTATION_PROJET.pdf`

4.4 UX / UI

- Respect des principes Material Design
- Navigation simple et intuitive
- Messages d'erreur clairs (login, réservation, réseau, etc.)

4.5 Compatibilité

- Plateformes :
  - Android (via `/android`)
  - iOS (via `/ios`)
- Backend :
  - Node.js
  - MySQL

---

5. Architecture technique

5.1 Vue globale

- **Frontend** : Flutter – `frontend/`
- **Backend API** : Node.js + Express – `backend/`
- **Base de données** : MySQL – scripts et schéma dans `database/`

Communication : **HTTP/JSON** entre Flutter et l'API.

5.2 Modèle de données (simplifié)

Table `users`

- id (PK)
- name
- email (unique)
- password_hash
- role (`client`, `host`, `admin`)
- created_at

Table `menu_items`

- id (PK)
- name
- description
- price
- category (`starter`, `main`, `dessert`, `drink`)

Table `reservations`

- id (PK)
- user_id (FK → users.id)
- reservation_date
- reservation_time
- guests
- phone
- status (`pending`, `confirmed`, `rejected`, `cancelled`)
- created_at

---

6. Organisation du projet

6.1 Rôles dans l'équipe (exemple)

- **Nouhaila MOUKADDIME** - Gestion de projet, UI/UX, Authentification
- **Axel Colliaux** - Gestion des menus, Wireframes
- **Noureddine BEN SADOK** - Backend, API, Base de donnees
- **Ilias Abdelkader EZEROUALI** - Systeme de reservations

6.2 **Methodologie Agile**

Le projet suit une methodologie Agile avec :

- **Board Kanban** : GitHub Projects
- **User Stories** : Issues GitHub
- **Reviews** : Code reviews systematiques

---

7. Planning (à adapter)

- **Phase 1 – Conception**
  - Cahier des charges
  - Modèle de données (`init.sql`, `schema.png`)
  - Rédaction des user stories
- **Phase 2 – Mise en place technique**
  - Initialisation frontend Flutter
  - Initialisation backend Node.js
  - Mise en place de la BDD
- **Phase 3 – MVP**
  - Authentification simple
  - Affichage du menu
  - Création de réservation simple
- **Phase 4 – Fonctionnalités complètes**
  - Gestion des statuts

    - Mes réservations
    - Vue hôte / serveur
- **Phase 5 – Tests & doc**
    - Documentation API
    - Guide d'installation
    - PDF final de projet
- **Phase 6 – Démo**
    - Enregistrement de `demo/demo_video.mp4`

---

8. Livrables

- Code complet du projet dans le dépôt Git
- Documentation dans `docs/`
- Script SQL `database/init.sql`
- Schéma de la BDD (`database/schema.png`)
- Vidéo de démonstration (`demo/demo_video.mp4`)
