**PROMPT MAÎTRE — Application de gestion Restaurant/Lounge Bar (Cotonou)**

Tu es un architecte logiciel senior + lead full-stack. Construis une application complète de gestion pour un **restaurant lounge bar** nommé **RMS Lounge Bar** situé **à Cotonou (Bénin)**. L’objectif est un **MVP robuste**, modulaire, prêt à déployer, avec **coûts d’infrastructure AWS minimaux**.

Pour une approche plus agile, il faut me proposer une breakdown du projet en phases (MVP puis features avancées) et une estimation des coûts AWS mensuels basée sur l'architecture proposée.

**Contexte & Contraintes**

* Propriétaire du domaine : **grouperms.com** (registrar GoDaddy).
* Compte AWS disponible. Priorité **serverless & free-tier friendly**.
* Localisation : **FR** (français), devise **XOF (CFA)**, fuseau **Africa/Porto-Novo**.
* Cibles : web (desktop + mobile) + **PWA** (mode offline pour le POS).
* Sécurité & conformité basiques (authentification, journaux, rôles).

**Architecture cible (AWS)**

* **Front-end** : React + Vite + TypeScript, PWA (App Shell), TailwindCSS. Hébergement **S3 + CloudFront** (HTTPS, HTTP/2), gestion DNS via **Route 53** (ou CNAME depuis GoDaddy).
* **AuthN/AuthZ** : **Amazon Cognito** (Email + option OTP SMS plus tard), rôles/parcours : Admin, Manager, Barman/Serveuse, Caissier, Comptable, Invité (lecture).
* **Back-end** : **API Gateway** REST + **AWS Lambda** (TypeScript/Node 20 ou Python 3.12).
* **Données** : **DynamoDB** (single-table design si possible) + **DynamoDB Streams** (audit/CDC), **S3** (pièces justificatives/exports).
* **Messaging/Jobs** : **EventBridge** (tâches planifiées), **SNS** (notifications), **SES** (emails).
* **Observabilité** : **CloudWatch** (logs/metrics/alertes), **X-Ray** (traces).
* **IaC** : **AWS CDK** (TypeScript) ou **SAM**. Fournis scripts/stack prêts à déployer.
* **Coûts** : On-Demand + faible trafic. Optimise lectures/écritures DynamoDB, compresse/mettre en cache via CloudFront, évite services coûteux.

**Domaines Fonctionnels (MVP)**

1. **POS / Ventes**
   * Prise de commande rapide (bar/cuisine), gestion tables/emporter/livraison simple.
   * Tickets, remises, annulations contrôlées, réductions happy-hour configurables.
   * Paiements : espèces, **caisse**, **banque**, **comptes MoMo** (ex. MTN MoMo, Moov Money), carte (gateway externe future).
   * Émission reçus (impression/PDF), fermeture de caisse, rapports Z journaliers.
   * Mode **offline-first** (IndexedDB + queue + resynchronisation).
2. **Stock / Recettes**
   * Articles (boissons/plats), variantes, unités, prix.
   * **Fiches techniques/recettes (BOM)** : décrémentation de stock par vente.
   * Entrées (achats), sorties (pertes/casse), inventaires, alertes seuil.
   * Valorisation stock (dernier coût, coût moyen), **COGS**, marge brute.
3. **Finances & Trésorerie**
   * **Positions de trésorerie** : Caisse(s), Banque(s), **MoMo** (comptes mobiles).
   * **Charges & Frais** (fournisseurs, loyers, salaires, utilités…), pièces jointes (PDF/JPG).
   * **Investissements (CAPEX)**, catégories, amortissement simple (linéaire configurable).
   * Écritures/Journal minimal, export CSV/Excel pour compta externe.
   * Rapports : CA, COGS, marge brute, EBITDA simplifié, flux de trésorerie.
4. **Ressources humaines**
   * Fiches employés, rôles, droits, plannings, pointages (timesheets) basiques.
   * Suivi primes/avances/retenues (export paie CSV).
   * Historique des accès/actions (audit).
5. **Paramétrage**
   * Taxes (TVA configurable), happy-hours, séries de numérotation, moyens de paiement, comptes de trésorerie.
   * Multi-site “plus tard” (prévoir clé siteId).

**Non-fonctionnel**

* **Sécurité** : IAM least-privilege, tokens JWT Cognito, validation côté client/serveur, chiffrement at-rest (DynamoDB KMS par défaut) et in-transit (HTTPS).
* **Qualité** : tests (unitaires/integration), ESLint/Prettier, GitHub Actions (CI), CD (optionnel).
* **Perf** : pagination, indexes GSI ciblés, cache CloudFront, images optimisées.
* **Accessibilité** : composants accessibles (WAI-ARIA), contraste suffisant.
* **Traçabilité** : audit basique (qui a fait quoi, quand).

**Modèle de Données (proposition single-table DynamoDB)**

* Table rmslounge\_core (PK/SK) avec entités typées :
  + ORG#<orgId> | SITE#<siteId>
  + USER#<userId>
  + ITEM#<itemId> (produit)
  + RECIPE#<itemId> (BOM)
  + STOCKMOVE#<moveId>
  + ORDER#<orderId> (POS)
  + PAYMENT#<paymentId>
  + CASHBOX#<cashboxId> (caisse) / BANK#<bankId> / MOMO#<momoId>
  + EXPENSE#<expenseId> (charges/frais) + justificatifs S3
  + ASSET#<assetId> (investissements) + planning amortissement
  + GSI1 par entityType, GSI2 par siteId, GSI3 par date, adaptées aux principales requêtes.
* Fournis schémas TypeScript (zod) pour validation.

**API (exemples d’endpoints REST)**

* POST /auth/signup|login (Cognito hosted UI conseillé)
* GET/POST /items, GET /items/:id, POST /items/:id/recipe
* POST /orders, GET /orders?date=, POST /orders/:id/payments
* POST /stock/moves, GET /stock/level?itemId=
* POST /cash/close, GET /cash/positions
* POST /expenses, GET /expenses?month=
* POST /assets, GET /assets, GET /kpis?from=&to=
* Rate limiting/API keys si usage public (sinon Cognito authorizer).

**UI/UX (pages clés)**

* Tableau de bord : CA du jour, tickets, marge, ventes top N, alertes stock, positions trésorerie (Caisse/Banque/MoMo).
* POS (rapide, tactile, favoris, code-barres optionnel).
* Produits & Recettes, Stocks & Inventaires.
* Trésorerie (caisse/banque/MoMo), Charges/Frais, Investissements.
* RH & Accès, Paramètres, Journal/Audit.
* Exports CSV/Excel, impressions PDF (tickets, rapports).

**Intégrations de paiement (à préparer)**

* Placeholder pour intégrations **Mobile Money** locales et passerelles (FedaPay/Stripe/Paystack, etc.). Fournir interface « adapter » pour greffer un provider sans refactor.

**Livrables attendus**

1. **Monorepo** (pnpm workspaces ou npm) avec :
   * /frontend (React TS, PWA, Tailwind, routing, state manager léger).
   * /backend (Lambdas + API Gateway, handlers, schémas zod).
   * /infra (CDK TypeScript ou SAM) : stacks S3/CloudFront/Route53 (option), Cognito, API Gateway, Lambdas, DynamoDB, SES/SNS/EventBridge, CloudWatch.
2. **Fichiers IaC** prêts à déployer, outputs (URLs, pool IDs).
3. **Script de seed** : création d’un site RMS-Lounge-Bar, produits tests (bières, vins, cocktails, plats), caisse “Bar”, banque “BOA”, MoMo “MTN”, utilisateurs démo (admin/manager/serveuse/caissier).
4. **Tests** (vitest/jest), **lint**, **GitHub Actions** CI.
5. **Docs** : README déploiement pas-à-pas (AWS CLI/CDK init), variables à renseigner (domain, ACM cert, SES verify), guide d’utilisation MVP, matrice des rôles/droits.
6. **OpenAPI 3.1** du backend, swagger.json + page Swagger UI (protégée).
7. **Mesures de coût** et check-list d’optimisation (DynamoDB on-demand, TTL logs, tailles objets S3, minification, mise en cache, alarmes free-tier).

**Exemples d’acceptation (MVP)**

* Créer produits + recette (BOM), vendre via POS, encaisser (espèces, MoMo simulé), décrémenter stock, clôturer caisse, voir CA/COGS/marge jour, saisir une charge, visualiser positions de trésorerie (Caisse/Banque/MoMo), exporter CSV mensuel.
* Offline POS : créer 3 tickets sans réseau, synchroniser à la reconnexion.
* Rôles : un caissier ne peut pas modifier les recettes ni voir salaires.

Génère directement :

* Le **squelette du code**, les **stacks IaC**, le **README** détaillé, l’**OpenAPI**, et **quelques pages front** (Dashboard, POS, Produits, Trésorerie) pour un **MVP fonctionnel**.

**PHASES DU PROJET PAR MODULES (par étapes)**

1. **Infra AWS & Déploiement**

Génère l’infrastructure AWS complète via **AWS CDK TypeScript** : S3 (hosting), CloudFront (SSL/TLS), Route 53 record (ou doc pour CNAME GoDaddy), Cognito (User Pool/Client/Groups), API Gateway (JWT authorizer), DynamoDB (single-table), Lambdas (Node 20), EventBridge, SES (checklist verify), SNS, CloudWatch alarms. Fournis le code CDK, variables, et un README “pas-à-pas” pour le déploiement.

1. **Backend + OpenAPI**

Crée le backend (Node 20 + TypeScript + zod) avec endpoints décrits ci-dessus, répertoires clairs (handlers/, services/, models/), validation, mapping DynamoDB, tests jest, et **OpenAPI 3.1** (swagger.json + route Swagger UI sécurisée).

1. **Front-end PWA**

Crée un front React + Vite + TS + Tailwind. Intègre Cognito (Hosted UI), gestion état (Zustand/Redux Toolkit léger), pages Dashboard, POS (offline-first avec IndexedDB + stratégie de sync), Produits/Recettes, Stocks, Trésorerie, Charges. Ajoute manifest.json, service worker, icônes.

1. **Stock & Recettes**

Implémente le module inventaire : produits, unités, recettes (BOM), mouvements (achats, pertes, inventaires), calcul stock courant, seuils et alertes. Fournis UI + endpoints + tests.

1. **Trésorerie & Finances**

Implémente caisses/banques/MoMo, écritures, clôtures, charges/frais, investissements (CAPEX + amortissement linéaire). Rapports CA/COGS/marge, positions de trésorerie, exports CSV/Excel.

1. **POS Offline-first**

Implémente POS tactile rapide avec favoris, splitting simple, remises, tickets parking. Mode offline (queue ops), résolution des conflits, synchro fiable.

1. **Sécurité & Rôles**

Paramètre Cognito groups/claims, middlewares d’autorisations, journal d’audit des actions sensibles. Fournis matrice rôles/droits et tests d’accès.