📌 Backlog de Cadrage Audit — Arka (R1 → M3)

🎯 Finalité

Établir un **backlog exhaustif et ultra détaillé** des attendus Arka, couvrant les lots 1 à 3. Chaque User Story est accompagnée de son **Definition of Done (DoD)** complet (critères G/W/T, branchements, logs, perf, accessibilité). Objectif : éliminer toute ambiguïté et éviter que les agents improvisent.

🗂️ Lot 1 — Fondations Console

Epic : Authentification & Accès

- **US1 — Connexion utilisateur**
  En tant qu'utilisateur, je peux me connecter via `/login`.
  DoD :

  1. Page `/login` affichant champs `email` et `mot de passe`.
  2. Bouton "Se connecter" déclenche un POST `/api/auth/login`.
  3. Réponses API : 200 → redirect `/console`; 401 → message d'erreur clair; 400 → validation côté client (champ manquant/invalide).
  4. Logs JSON : `{ts, user, status, route}` créés à chaque tentative.
  5. Tests QA Given/When/Then PASS :
     - Given identifiants valides → When login → Then redirection console.
     - Given identifiants invalides → When login → Then message erreur.

- **US2 — Rôles utilisateurs (RBAC)**
  En tant qu'utilisateur, mon rôle (Viewer, Operator, Owner) conditionne mes accès.
  DoD :

1. Middleware back vérifie JWT + rôle.
2. UI console affiche rôle actif (badge ou indication claire).
3. Routes `/console/*` protégées selon rôle (403 si accès interdit).
4. Tests QA : accès autorisé/interdit validés pour chaque rôle.

Epic : Multi-Clients

- **US3 — Gestion multi-tenants**
  En tant qu'owner, je peux gérer plusieurs espaces clients.
  DoD :
  1. Structure DB supportant plusieurs clients.
  2. API expose champ `client_id` obligatoire.
  3. Console : sélecteur de client affiché si rôle Owner.
  4. Tests QA : changement de client = changement contexte données.

Epic : Observabilité de base

- **US4 — Health-check**
  En tant qu'opérateur, je peux interroger `/api/health`.
  DoD :
  1. API `/api/health` répond 200 avec `{status: "ok", ts}`.
  2. Logs JSON créés pour chaque appel.
  3. Test QA validant réponse ≤100ms.
- **US5 — Logs initiaux**
  En tant qu'opérateur, je peux consulter logs d'accès.
  DoD :
  1. Format JSON unique : `{ts, level, msg, route, status}`.
  2. Logs présents sur login, console, health.

3. Test QA vérifiant création log par action.

🗂️ Lot 2 — UI & Routage

Epic : Routage & Navigation

- **US6 — Routage de base**
  En tant qu'utilisateur, je peux naviguer entre `/`, `/login`, `/console`.
  DoD :

  1. Routes définies et accessibles.
  2. 404 affichée pour toute route inexistante.
  3. Test QA : navigation valide/404 vérifiée.

Epic : UI/UX

- **US7 — Composants UI de base**
  En tant qu'utilisateur, je peux utiliser une UI cohérente.
  DoD :

  1. Composants (boutons, inputs, cards, modales) disponibles et stylés.
  2. Accessibilité AA vérifiée (contraste, aria-labels).
  3. Test QA automatisé sur composants.

- **US8 — Design System**
  En tant que designer/dev, je peux utiliser un design system standardisé.
  DoD :

  1. Tokens de style centralisés (couleurs, typo, spacing).
  2. Documentation Codex-ready disponible (`spec-integration.md`).
  3. Validation QA intégration graphique PASS.

Epic : Storybook

- **US9 — Storybook UI**
  En tant qu'équipe dev, je peux consulter les composants UI dans Storybook.
  DoD :

    1. Chaque composant exposé dans Storybook.
    2. Story incluant variantes principales.
    3. QA validant la conformité Storybook / design system.

🗂️ Lot 3 — Qualité & Gouvernance

Epic : Qualité Logicielle

- **US10 — Tests automatiques**
  En tant que QA, je dispose de tests couvrant login, navigation, RBAC.
  DoD :

    1. Tests unitaires sur fonctions clés.
    2. Tests E2E (login → console).
    3. Rapport PASS/FAIL livré par QA.

- **US11 — Accessibilité**
  En tant qu'utilisateur, je bénéficie d'une UI AA.
  DoD :

    1. Vérification contraste couleurs.
    2. Navigation clavier.
    3. Tests automatisés axe-core ou équivalent.

Epic : Performance

- **US12 — Temps de chargement**
  En tant qu'utilisateur, je bénéficie d'une console réactive.
  DoD :

1. LCP ≤ 2.5s sur `/` et `/login`.
2. TTI ≤ 2s sur `/console`.
3. Rapport perf validé QA.

Epic : Sécurité

- **US13 — Validation JWT**
  En tant que système, je rejette les requêtes invalides.
  DoD :

  1. JWT requis pour toute route protégée.
  2. Requêtes invalides → 401/403.
  3. Logs JSON créés pour chaque rejet.
  4. Tests QA validés.

✅ Synthèse

- **Lot 1** : Auth (login + RBAC), Multi-clients, Logs, Health-check.
- **Lot 2** : Routage, UI de base, Design system, Storybook.
- **Lot 3** : QA tests, Accessibilité, Perf, Sécurité JWT.

📌 Usage

- Ce backlog **ultra cadré** est la référence unique pour audit Codex.
- Toute US non respectée intégralement = **KO en audit**.
- Aucune improvisation tolérée : DoD = vérité absolue.