CAPSTONES — Projets signatures (Studio-grade)

Cahiers des charges + rubriques + packaging

RBK 2.0

Contents

1 Philosophie “standard studio” + checklist non négociable

Principe. Un capstone RBK est un livrable **utilisable**, pas un prototype. Il comprend : code, tests, documentation, menaces, observabilité, runbook, release et démonstration reproductible.

Non négociables (Studio-grade)

- Spec + critères d’acceptation mesurables.
- Threat model + security checklist.
- Tests : unit + intégration + négatifs (et fuzz si pertinent).
- CI : lint + tests + rapport.
- Observabilité : logs/métriques + alerting minimal.
- Runbook incidents : symptômes → diagnostic → action.
- Demo rejouable : script + scénario + données.

Catégorie	Preuve attendue
Spec & DoD	document + checklist acceptance (15–25 items)
Sécurité	threat model + findings + correctifs testés
Tests	suite complète + logs CI + cas négatifs
Docs	architecture + API + schémas + guide déploiement
Ops	runbook + dashboard + signaux/alertes
Release	tags + changelog + address book / manifest

Table 1: Checklist studio-grade (résumé)

2 Gabarit unique A→J (à appliquer à chaque capstone)

A) Problème & Contexte

B) Personas & User Stories (min 6)

C) Architecture cible (onchain/offchain/indexer/UI/obs)

D) Spécification Smart Contracts (comptes, instr, events, invariants)

E) Threat model (STRIDE simplifié) + hypothèses

F) Plan de tests (unit/int/e2e/fuzz) + seuils

G) Performance budget (latence, CU/gas, RPC strategy)

H) Observabilité & Runbook

I) Critères d'acceptation (checklist mesurable)

J) Livrables (repo structure + docs + demo + audit report)

3 Capstone 1 : Wallet & Transaction Reliability Pack

A) Problème & Contexte

Les dApps Web3 échouent souvent par **instabilité transactionnelle** : erreurs RPC, timeouts, confirmations incertaines, UX dégradée, et absence d'outillage de diagnostic. L'objectif est de produire un "reliability pack" réutilisable : state machine transaction, taxonomie d'erreurs, stratégies de retry, instrumentation et dashboard.

B) Personas & User stories

- **User** : "En tant qu'utilisateur, je veux comprendre pourquoi ma transaction a échoué."
- **Support** : "En tant que support, je veux un diagnostic rapide (cause probable + action)."
- **Dev** : "En tant que dev, je veux instrumenter la tx lifecycle et mesurer les erreurs."
- **Ops** : "Je veux détecter une hausse d'échecs RPC et déclencher une mitigation."
- **PM** : "Je veux suivre le taux de succès et la latence de confirmation."
- **Partner** : "Je veux un mode verification' pour reproduire un incident."

C) Architecture cible

Front (state machine tx) + couche RPC strategy (providers, fallback) + telemetry (events) + dashboard + playbooks support.

D) Spécification (composants)

- **Tx State Machine** : created → signed → sent → confirmed → finalized (ou failed).
- **Error taxonomy** : erreurs wallet, RPC, simulation, blockhash, compute, signature.
- **Retry policy** : règles déterministes, backoff, limite, passage provider.

E) Threat model (simplifié)

- Spoofing : faux provider / réponses falsifiées → mitigation : allowlist providers, signatures.
- DoS : surcharges RPC → mitigation : fallback + cache + rate limit.
- Repudiation : logs absents → mitigation : trace IDs et journaux horodatés.

F) Plan de tests

Unit tests (state transitions) + intégration (provider failover) + tests "chaos RPC" (timeouts) + tests de messages UX.

G) Performance budget

- Latence UI : affichage état $\leq$ 200 ms après événement.
- Stratégie confirmations : seuil configurable ; fallback si non confirmé.

H) Observabilité & Runbook

Métriques : success rate, fail types, provider errors, confirm latency. Runbook : hausse des fails → switch provider → degrade features → informer user.

I) Critères d'acceptation

Voir Table 3 (20 items).

J) Livrables

Repo + docs + dashboard + demo script + mini "support guide".

Erreur	Cause probable	Mitigation / UX
Timeout RPC	provider saturé	fallback provider + message "réessayer"
Blockhash expired	tx trop lente	re-sign + refresh blockhash
Simulation failed	état invalide	expliquer précondition + lien docs
Signature rejected	wallet / user	demander re-sign + check wallet
Compute limit	tx trop lourde	proposer simplification / split tx

Table 2: Taxonomie erreurs Wallet/RPC (extrait)

Acceptance criteria (Capstone 1) — 20 items mesurables (exemples)
1) State machine couvre 100% des transitions prévues + tests transitions invalides.
2) 12 types d'erreurs minimum catégorisés, chacun avec mitigation + message UX.
3) Fallback provider fonctionnel (démo : provider A down → provider B).
4) Dashboard : success rate, latency, top errors, provider errors.
5) Runbook : au moins 3 incidents simulés + résolution.
6) Demo script rejouable depuis fresh clone.

Table 3: Acceptance criteria Capstone 1 (à compléter jusqu'à 20 items)

4 Capstone 2 : Tokenization & Admin Control Center (RWA/Token-2022)

A) Problème & Contexte

Les projets de tokenization échouent par manque de **contrôles admin**, **piste d'audit**, politiques (RBAC), et procédures (approbation, exécution, rollback). Ce capstone construit un "Control Center" : interface admin + policies + audit trail + vérification.

B) Personas & user stories

Admin, compliance, ops, support, auditor, partner.

C) Architecture cible

Admin UI → API/Policy engine → indexer/audit store → smart contracts → dashboard.

D) Spec smart contracts (résumé)

Roles, permissions, events d'audit, state machine actions (mint/burn/freeze/whitelist).

E) Threat model (simplifié)

Elevation of privilege (RBAC mal conçu) ; repudiation (audit logs absents) ; tampering (policy contournée).

F) Tests

RBAC tests (positifs/négatifs), tests audit trail (events), tests rollback.

G) Perf budget

Temps d'exécution policy, latence admin UI, intégrité audit.

H) Observabilité & runbook

Alertes : actions admin inhabituelles ; anomalies permissions ; freeze/unfreeze.

I) Acceptance criteria

Inclure RBAC matrix + audit trail schema + PRR.

J) Livrables

Repo + docs compliance-friendly + démo + audit report.

Rôle	Permissions	Risque si mal configuré
Issuer	mint/burn, set policy	inflation / abus
Compliance	whitelist/freeze	censure/erreurs de blocage
Operator	exécuter jobs	opérations frauduleuses
Auditor	read-only + exports	fuite d'infos

Table 4: RBAC matrix (extrait) — Capstone 2

Event d'audit	Champs minimaux
PolicyUpdated	who, when, diff, rationale
MintExecuted	who, amount, recipient, policy-id
FreezeAction	who, target, reason, duration

Table 5: Audit trail schema (extrait)

5 Capstone 3 : Digital Assets & Utility Ecosystem (NFT/Gating)

A) Problème & Contexte

Les NFTs sans utilité réelle sont fragiles. Ce capstone impose une utilité **gated**, vérifiable, performante : "My Assets", accès features, perks, dashboards, indexation, et UX stable.

B) Personas & user stories

Holder, new user, support, partner, devrel, ops.

C) Architecture

Wallet connect → gating checks (on-chain + cache) → unlock features → telemetry. Indexer : events → DB → cache → API.

D) Spec

Tiers, règles de gating, events, invalidations cache, policy updates.

E) Threat model

Spoof gating, cache poisoning, replay, DoS sur indexer.

F) Tests

Unit (rules), integration (indexer), e2e (unlock), perf (cache).

G) Perf budget

Latence gating $\leq$ 300ms (cible), taux cache hit, fallback on-chain.

H) Observabilité

Métriques gating latency, unlock success, indexer lag.

I) Acceptance criteria

Matrice utility + 15+ critères.

J) Livrables

Repo + docs + demo + audit.

Tier NFT	Feature	Check	UX attendu
Bronze	accès contenu A	on-chain + cache	unlock immédiat
Silver	accès bounties	check + signature	écran “eligible”
Gold	priority support	tier check	badge UI + SLA

Table 6: Utility mapping (extrait) — Capstone 3

6 Grille d’évaluation 100 points (jury/audit)

Catégorie	Poids	Critères
Sécurité & menaces	25	threat model + tests négatifs + corrections
Tests & qualité	20	suite complète + CI + coverage utile
Architecture & scalabilité	15	indexer/caching, dépendances, ADR
Observabilité & runbook	15	métriques, alertes, incidents simulés
UX & robustesse wallet/RPC	10	messages clairs, retries, state machine
Docs & audit report	15	package complet et vérifiable

Table 7: Rubrique capstones (100 points)

7 Package final (repo + docs + demo + audit)

Item	Contenu minimal
Repo structure	/programs /app /tests /scripts /docs /monitoring
Docs	architecture, API, threat model, runbook, deployment/rollback
Audit report	10 pages min : findings, sévérité, correctifs, preuves
Observabilité	dashboard + règles d’alerting + playbooks
Demo	vidéo + script live + scénario

Table 8: Package final (DoD capstone)

Figure 1: Packaging pipeline : build → test → audit doc → release → demo