Chapitre 14. La plateforme MANARA et son évaluation experte

VN-Ch14-1 : Pile technique, modèle de données et fonctionnement hors ligne

Complément du chapitre 14, section 14.2.1

14.2.2. Les choix de pile et leur justification

Le tableau VN-41 liste les choix de pile avec les versions épinglées dans les fichiers de configuration des paquets, dont nous avons vérifié la présence dans le fichier de verrouillage du gestionnaire de paquets (75 dépendances à version exacte, aucune absente). La justification est celle que la conception consigne, ou celle de la spécification de conception lorsque le choix y est expliqué ; nous ne reconstruisons pas de motif qui n'y figure pas.

Tableau VN-41 : Pile technique du dispositif MANARA, versions épinglées et justification consignée.

Couche	Choix et version	Justification consignée (source)
Code source et outillage	Monorepo pnpm 10.28.0 ; Node 24 en CI (version minimale 24.7) ; TypeScript 5.9.3	Nouveau code source, rien repris de l'ancien code ; versions épinglées, une commande de contrôle unique
Schémas partagés	Zod 4.6.7 dans le paquet schéma	Une seule définition du contenu, des événements et des objets d'échange, partagée entre interface, serveur et build de contenu
Interface	React 19.3.0, Vite 8.0.16	Pile de l'existant, déjà maîtrisée (spécification de conception, 10.1)
Routage	wouter 3.11.0	Option légère choisie par le concepteur contre un routeur plus lourd, au nom du budget de 250 Ko
Données et langues	TanStack Query 5.103.1 ; i18next 26.4.2 et react-i18next 17.0.14	Choix de la spécification de conception
Style	Tailwind 4.3.3, jetons générés préfixés --mn-	Outil de build seulement, seul le CSS généré est livré ; alternative écartée : CSS sans utilitaires
Hors ligne	Service worker écrit à la main, IndexedDB sans bibliothèque	Aucune dépendance, une soixantaine de lignes lisibles ; l'API n'est jamais mise en cache
Contenu	MDX, JSON et YAML ; remark-mdx 3.1.1, unified 11.0.5, yaml 2.9.1 ; validation Zod au build	Contenu versionné avec le code : une version gelée est une étiquette et une empreinte de paquet (chapitre 12, 12.5.1)
Serveur	Express 5.2.1, regroupé par esbuild 0.28.2	Continuité avec l'existant (spécification, 10.1) ; esbuild parce que la compilation TypeScript en NodeNext exigerait des extensions dans tout le monorepo
Accès aux données	Prisma 7.10.0, adaptateur pg 7.10.0, pg 8.23.0	Version stable choisie car l'étiquette courante désignait une préversion ; SQL écrit à la main écarté
Base de données	PostgreSQL 16 (image alpine)	Écritures concurrentes de plusieurs classes et exports volumineux (spécification, 10.1)

Couche	Choix et version	Justification consignée (source)
Authentification	Sessions par cookie ; argon2id natif de Node ; second facteur TOTP pour chercheur et administrateur	Révocation simple, aucune dépendance de hachage
Proxy et conteneurs	Caddy 2, Docker Compose, images node:24-alpine, postgres:16-alpine, caddy:2-alpine	Caddy sert le statique et relaie /api, une brique de moins
Tests	Vitest 5.0.1, Testing Library 16.3.3, jsdom 30.1.0, supertest 7.2.2, Playwright 1.63.0, axe-core 4.13.0	Parcours complets sur le Chrome du poste, audit d'accessibilité à chaque écran
Qualité	ESLint 10.10.0, Prettier 3.9.8, Lighthouse 13.5.0	Contrôles bloquants en CI ; Lighthouse en tâche à part
Assistant	SDK Anthropic 1.3.0, dans l'API seulement	Service serveur appelé à la demande

Note. La colonne de justification rapporte le motif consigné par la conception ; elle ne constitue pas une évaluation de ce motif.

Le tableau VN-41 montre un trait de la pile qui découle des principes du chapitre 12 : la plupart des choix minimisent la dépendance. Le hors ligne n'utilise ni Workbox ni bibliothèque d'accès à IndexedDB ; le hachage des mots de passe n'appelle aucune dépendance ; le routage est le plus léger des routeurs considérés. Chacun de ces choix répond au budget de sobriété autant qu'à une préférence de maîtrise, et la justification le dit explicitement quand c'est le cas.

14.2.3. Le modèle de données et les routes

Le schéma Prisma compte 27 modèles et 13 migrations. Les rôles de la base sont l'élève, l'enseignant, le chercheur, l'administrateur et l'expert, et les niveaux sont 3AC et 2BAC. Les modèles se répartissent en cinq groupes : les comptes et les classes, le contenu et ses versions, la progression et les résultats, les traces et les journaux, et les cours d'enseignants avec leurs versions, publications et programmations. La route de l'API est organisée en onze modules (administration, assistant, classes, cours, exports, journaux, résultats, séance, suivi, synchronisation, tableau) auxquels s'ajoute l'authentification. Une fonction d'autorisation unique, qui consulte une matrice rôle par ressource, garde chaque route ; dans cette matrice, le rôle d'expert n'a aucun droit, ce que le code commente comme un état d'attente de la version d'évaluation experte.

14.2.4. Hors ligne et journalisation

Le chapitre 12 a posé le fonctionnement sans réseau comme état de référence et la conservation de l'horodatage d'origine comme condition de validité des traces. Le code source contient les composants correspondants : un service worker généré au build avec la liste exacte des fichiers

de la coquille, un téléchargement explicite des paquets de chapitre, un gestionnaire de traces qui met les événements en file locale, une synchronisation par lots, un module d'horodatage local et des identifiants d'événement ordonnés dans le temps. Le schéma des événements compte onze types : les sept de la spécification des traces, auxquels s'ajoutent la séance de projection, la séance composée, la réponse à la sonde et la révision.

Nous avons rejoué le parcours « mode avion » du code source sur une construction de production, avec le service worker réel et une API simulée par le navigateur de test. Il s'est déroulé en 8,5 secondes de test (13,6 secondes avec la construction) et a réussi. L'élève se connecte et télécharge le chapitre de la digestion, le service worker prend le contrôle de la page ; le réseau est coupé ; l'application et la fiche du chapitre se rouvrent, l'élève répond aux sept questions de la sonde et passe à l'activité suivante sans qu'aucun événement ne parte vers le serveur ; l'indicateur d'état affiche alors un nombre d'événements en attente. Le réseau revenu, la file part : les événements de réponse à la sonde portent un horodatage compris dans la fenêtre de la coupure, à une seconde près, leurs identifiants sont tous distincts, l'indicateur disparaît et le navigateur n'a relevé aucune violation de la politique de sécurité de contenu. Ce rejeu établit le comportement du client sur ce scénario ; il ne dit rien du serveur réel, de l'appareil réel ni d'un parcours plus long. Reste à dire dans quel environnement ce code est construit, testé et mis en ligne.

VN-Ch14-2 : Environnement de développement et chaîne d'intégration et de livraison

Complément du chapitre 14, section 14.2.1

14.3. L'environnement de développement et de déploiement

14.3.1. Le poste de développement

Le développement se fait sur un poste de travail, avec les outils de la pile décrite plus haut (Node 24.7 au moins, pnpm 10.28.0). Les commandes principales sont une commande de contrôle (pnpm check : analyse statique, types, tests unitaires, construction et budget de poids), des parcours de bout en bout sur le navigateur Chrome, des tests d'intégration sur PostgreSQL et une pile complète avec jeu de données fictif.