CacheLab-CCT

1- langage choisi : Node.js , Type Script

Justification :

TypeScript : offre un typage statique permettant d'éviter les erreurs fréquentes lors de la manipulation des données

Node.js : est performant pour créer des **APIs** rapides et légéres, ce qui correspond aux besoins du projet

écosystéme riche , documentation abondante et large communauté

2. Architecture applicative

Composants :

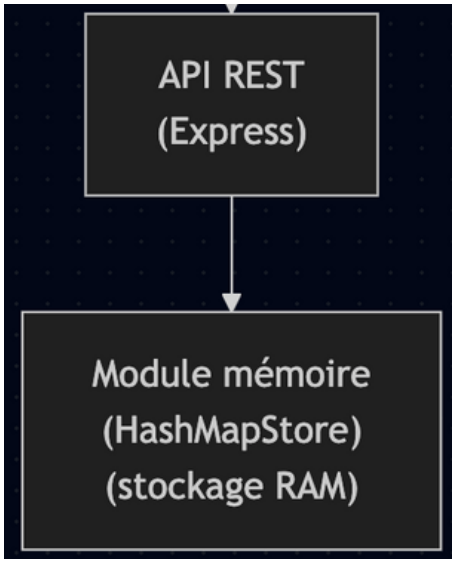
1- Serveur REST :

Expose des endpoints standards (**GET-POST-PUT-DELETE**)

JSON seulement, pas de rendu HTML

Authentification viaa clé API

Validation sticte des clés



2-Module de stockage en mémoire

Structure de données : table de hachage custom (style hashmap)

Stocke les paires ‘key-value’ directement en RAM

Construit uniquement avec :

tableaux natifs - objets simples-aucun usage de Map/Set, find,filter,reduce,ect...

3-Logging/Monitoring

Module léger
Chaque requête est tracée
Utile en cas de diagnostic ou d’optimisation

4-Middlewares de sécurité
auth.ts : vérifie la clé API
validateKey.ts : valide la structure et la longueur de la clé
errorHandler.ts : capture les erreurs globales pour éviter les crashs
requestLogger.ts : log des reqêtes

3- Structure de données retenue

Table de hachage avec chaînage (hashmap)

Objectif : O(1) en moyenne pour créer/lire/mettre a jour/supprimer

Structure interne :
nuckets : Entry()() → un tableau de “seaux”
Chaque “bucket” contient une list d’objets (key-value)
Fonction de hachage personnalisée :
Transforme une châine de caractéres ‘clé’ en numéro
Indexe ce numéro dans buckets via hash % capacity

Chaînage :
Si deux clés ont le même bucket → elles sont stockées côte a côte dans le tanleau

Par ce que :
Simple, efficace, proche du fonctionnement de Redis
Adapté au cahier des charges : mémoire +vitesse + simplicité
Permet de stocker n'importe quel type de donnée sous forme de valeur JSON

4- Endpoints API détaillés

	Methode	Endpoint	Description	Body	Réponse	
	---	---	---	---	---	
	POST	/keys	🔹 Créer une	{ "key":	201 created	
	GET	/keys/:key	🔹	(rien)	200 { key,	
	PUT	/keys/:key	🔹 Modifier	{ "value": 42	200 updated	
	DELETE	/keys/:key	🔹	(rien)	200 deleted	
	GET	/keys	🔹 Lister	(rien)	200 { keys:	

5- Mesures de sécurité

Authentification : clé API via header x-api-key
Validation : Regex sur les clés, taille max configurable
Gestion d'erreurs : Middleware global
Données sensibles : rien stocké de facon persistante - tout est en RAM