

CacheLab.

Contexte

Vous êtes une équipe de développeurs backend fraîchement recrutée par **CacheLab**, une startup en pleine croissance spécialisée dans les solutions de performance web.

L'entreprise vient de signer un contrat majeur avec plusieurs clients e-commerce qui font face à des problèmes de lenteur sur leurs sites pendant les pics de trafic (Black Friday, soldes...). Ces sites utilisent des bases de données relationnelles traditionnelles qui ne peuvent pas gérer les millions de requêtes par seconde.

Votre mission : Développer **de zéro votre propre système de cache clé/valeur** en vous inspirant des principes qui ont fait le succès de Redis. Ce système permettra à vos clients de stocker temporairement en mémoire les données les plus consultées (prix des produits, informations utilisateurs, paniers d'achat...) et ainsi réduire drastiquement la charge sur leurs bases de données principales.

L'objectif est de comprendre comment fonctionne ce type de base de données en la développant vous-même.



Le CTO vous a donné **2 semaines** pour livrer un **MVP (Minimum Viable Product)** fonctionnel qui sera testé avec le premier client pilote.

Phase 1: Phase de recherche

Avant de coder, vous devez comprendre comment Redis fonctionne et analyser le marché :

1. Veille sur les bases de données clé/valeur

- Qu'est-ce qu'une base de données clé/valeur et en quoi diffère-t-elle d'une base SQL ?
- Comparatif des solutions existantes
- Cas d'usage concrets

2. Analyse de Redis comme référence

- Comment Redis stocke les données en mémoire (structures de données utilisées) ?
- Pourquoi est-il si rapide ?
- Types de données supportés ?
- Mécanismes de persistance des données

3. Architecture et performance

- Comment fonctionne le stockage en mémoire vs sur disque ,
- Trade-offs : vitesse vs persistance, mémoire vs capacité ...



4. Aspects écoresponsables et sécurité

- Impact environnemental des serveurs en mémoire (consommation RAM)
- Bonnes pratiques de sécurité pour les API REST
- Conformité RGPD (si données personnelles dans le cache)

Phase 2 : Phase de développement

Commencer par choisir un des deux langages suivants pour votre implémentation : Node.js (JavaScript/TypeScript) ou Go. Justifiez ce choix dans votre cahier des charges techniques en argumentant sur les critères de performance, de facilité de développement et d'écosystème disponible.

Créez de zéro une structure de données pour stocker les paires clé/valeur.

Votre système doit :

- Supporter les opérations CRUD (Create, Read, Update, Delete)
- Gérer la mémoire efficacement
- Être optimisé pour la recherche rapide

Au minimum, les endpoints suivants doivent être implémentés :

Méthode	Endpoint	Description
POST	/keys	Créer une nouvelle clé/valeur
GET	/keys/:key	Récupérer la valeur d'une clé



PUT	/keys/:key	Modifier la valeur d'une clé existante
DELETE	/keys/:key	Supprimer une clé
GET	/keys	Lister toutes les clés (optionnel)

Vous devez implémenter et justifier le choix d'au moins un des algorithmes suivants :

- HashMap/Table de Hachage (recommandé pour les performances $O(1)$)
- Recherche dichotomique (sur tableau trié)

Expliquez dans votre documentation pourquoi vous avez choisi cet algorithme et quels sont ses avantages pour CacheLab.

Compétences visées

- Conduire une veille technologique et réglementaire dans une démarche d'éco-conception et de sécurité
- Synthétiser les données de veille en validant leur fiabilité et en faire une restitution compréhensible aux acteurs du projet
- Rédiger une note de cadrage précisant la démarche, les objectifs et cadrant délais, budget, ressources et qualité
- Concevoir une architecture applicative en analysant les besoins fonctionnels et les exigences de sécurité



- Justifier l'utilisation de patterns pour garantir une architecture modulaire, réutilisable et maintenable
- Recommander un environnement informatique éco-responsable en garantissant la cohérence du système
- Développer des applications métiers sécurisées en optimisant l'efficacité des développeurs

Rendu

Il est attendu les documents suivants :

Un **cahier des charges fonctionnel** avec les éléments suivants :

- Les besoins du client (performances attendues, cas d'usage)
- Les fonctionnalités prioritaires (CRUD sur clés/valeurs)
- Les contraintes techniques (temps de réponse, sécurité)
- La spécialisation choisie et ses implications fonctionnelles

Un **cahier des charges techniques** avec les éléments suivants :

- Langage choisi : Node.js/TypeScript OU Go (justification argumentée)
- Architecture applicative : Serveur API REST, Module de stockage en mémoire, Système de logging/monitoring, Composants spécifiques à la spécialisation.
- Structure de données retenue : HashMap ou autre (justification),
- Endpoints API détaillés avec exemples de requêtes/réponses
- Mesures de sécurité : validation des entrées, gestion des erreurs, authentification



Une **note de cadrage** et planification : décomposition en tâches, répartition des rôles dans l'équipe, estimation des délais, identification des risques (bugs, performances insuffisantes...)

Le projet est à rendre sur votre github :

<https://github.com/prenom-nom/CacheLab>

Base de connaissances

- [Comprendre la base de données en mémoire](#)
- [Redis: La database NoSQL la plus appréciée des développeurs](#)
- [Tutorial part-1: Key-value store](#)