Samit Sabah Adelyar
Techniques de l’informatique
groupe 04318

Projet 3 : Rapport de synthèse

Travail présenté à
M.Nicolas Payre
Département des Techniques de l’informatique
pour le cours
Exploration de nouvelles technologies

Cégep de Sherbrooke
06 décembre 2025

Introduction

Dans le cadre du projet 3 du cours Exploration de nouvelles technologies, j'ai choisi de comparer deux solutions modernes de Backend-as-a-Service : PocketBase, un service auto-hébergeable extrêmement léger reposant sur SQLite, et Supabase, une plateforme cloud complète bâtie autour de PostgreSQL. Bien que ces technologies remplissent des rôles similaires (authentification, API, base de données, realtime), elles reposent sur des architectures et des modèles d'hébergement très différents, ce qui influence fortement leurs performances, leurs coûts et leurs scénarios d'usage

Afin de rendre cette exploration concrète, j'ai développé PocketTasks, une application Svelte permettant de gérer une liste de tâches. Le prototype peut basculer dynamiquement entre les deux backends et mesurer la latence de chaque opération CRUD. Ce projet a également été l'occasion d'intégrer l'IA comme assistant technique, pédagogique et rédactionnel, afin d'en analyser l'apport réel dans un processus de développement logiciel.

Ce rapport présente une synthèse structurée de l'exploration : description des technologies, analyse comparative des performances, évaluation des coûts, rôle de l'IA et réflexion critique sur les limites et forces des solutions étudiées.

Description des technologies

Pocketbase

Description

PocketBase est un BaaS open-source écrit en Go qui se distingue par son extrême simplicité : l'ensemble du backend est contenu dans un seul exécutable, ce qui facilite son déploiement, notamment en self-hosting. Malgré sa légèreté, il inclut un tableau de bord administratif, un système d'authentification, une API REST, une API temps réel et un stockage de fichiers basique.

SQLite

PocketBase repose sur SQLite, une base de données embarquée. Celle-ci offre :

- Des performances très rapides en local
- Nne configuration quasi nulle
- Un fonctionnement monofichier très léger

Cependant, SQLite impose des limites structurelles :

- Single-writer lock : une seule écriture simultanée est possible
- absence de scalabilité horizontale
- Moins de fonctionnalités avancées (permissions complexes, monitoring, transactions distribuées)

Supabase

Description

Supabase est une plateforme BaaS hébergée dans le cloud. Elle regroupe plusieurs services intégrés : PostgREST, Auth, Realtime, Edge Functions, Storage API, etc. Son objectif est de fournir une alternative open-source complète à Firebase.

PostgreSQL

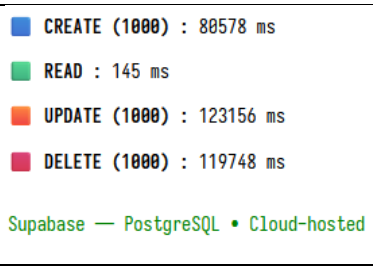
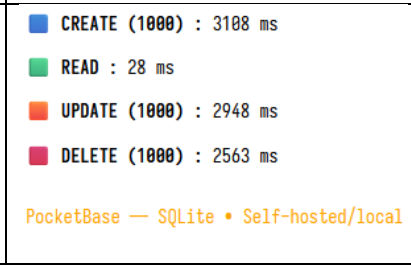
L'infrastructure repose sur PostgreSQL, un moteur relationnel robuste, performant et prêt pour des environnements de production à grande échelle. PostgreSQL permet :

- Transactions isolées
- Gestion lourde de concurrence
- Scalabilité verticale/horizontale
- Règles de sécurité avancées (RLS)
- Monitoring et outils d'administration matures

Analyse des performances

Les tests ont été réalisés via PocketTasks, en mesurant la latence de chaque opération CRUD et un benchmark intensif de 1000 opérations répétées.

Comparaison de performances

	Supabase	PocketBase
Ajouter une tâche	98 ms	26 ms
Modifier une tâche	94 ms	22 ms
Supprimer une tâche	83 ms	27 ms
Consulter toutes les tâches	98 ms	5 ms
Benchmark complet (créer,modifier et supprimer 1000 tâches)		

Explication des résultats

Les résultats du benchmark montrent un écart de performance très net entre PocketBase et Supabase, particulièrement lors des opérations lourdes (création, modification et suppression de 1000 tâches). PocketBase réalise ces opérations en quelques millisecondes (≈ 2500–3000 ms au total), alors que Supabase nécessite plusieurs dizaines de milliers de millisecondes pour les mêmes opérations (≈ 85 000–120 000 ms). Cet écart s'explique principalement par la différence d'architecture et de technologie sous-jacente : PocketBase repose sur SQLite, une base de données embarquée fonctionnant localement dans le même processus que le serveur, ce qui réduit considérablement la latence et le coût des transactions. De plus, le test a été effectué en environnement self-hosted/local, où les requêtes ne transitent pas par Internet.

À l'inverse, Supabase utilise PostgreSQL, un moteur robuste conçu pour la scalabilité et la gestion de nombreuses connexions simultanées, mais qui introduit une latence réseau inévitable puisqu'il s'agit d'un service cloud-hosted. Chaque opération CRUD doit traverser plusieurs couches (API REST, PostgREST, vérification des règles RLS, transactions PostgreSQL), ce qui alourdit mécaniquement le temps d'exécution même pour des charges modestes. Les opérations unitaires restent relativement rapides (≈ 83–98 ms), ce qui est cohérent avec une base distante, mais leur répétition dans une boucle de 1000 requêtes amplifie la latence cumulée.

Enfin, ces résultats mettent en évidence un aspect essentiel des deux plateformes : PocketBase excelle dans des scénarios locaux ou à petites charges grâce à sa simplicité et sa proximité avec la base de données, tandis que Supabase, bien que plus lent dans un benchmark intensif, offre une architecture mieux adaptée aux applications évolutives, multi-utilisateurs et nécessitant une sécurité avancée (RLS). Ainsi, la différence de performance brute observée ne reflète pas une supériorité absolue, mais plutôt les forces naturelles de chaque technologie selon leur contexte d'utilisation.

Comparaison dans le même contexte

Si PocketBase était hébergé sur le cloud comme Supabase (avec des outils comme Aws Lightsail ou Railway), ces technologies serait soumise aux mêmes contraintes concernant la latence réseau. Dans ce contexte, les performances seraient alors beaucoup plus proches l'une d'entre elle. PocketBase risquerait quand même d'avoir un avantage, car :

- Le Row-level Security de Supabase fait passer chaque requête par une vérification
- L'API REST n'est pas natif dans PostgreSQL
- PostgreSQL est naturellement plus robuste, mais aussi plus lourd

Cependant, dans un plus gros projet, Supabase sera plus efficace et plus stable en raison de sa meilleur scalabilité

Analyse des Coûts

L'analyse des coûts révèle deux philosophies radicalement opposées. PocketBase, entièrement gratuit et auto-hébergeable, ne nécessite qu'un serveur minimal coûtant entre 5 et 7 dollars par mois. C'est donc une solution extrêmement économique, mais qui transfère à l'équipe de développement l'ensemble des

responsabilités opérationnelles : gestion des sauvegardes, sécurité, monitoring et mises à jour. Supabase adopte une approche inverse : la plateforme est plus coûteuse (25 à 49 dollars par mois pour une application de production), mais elle fournit une infrastructure entièrement gérée, incluant la base PostgreSQL, le realtime, les backups, les certificats TLS, la scalabilité et le monitoring.

À long terme, PocketBase reste très avantageux pour des applications simples ou peu critiques, alors que Supabase devient plus rentable pour des projets nécessitant croissance, sécurité avancée, haute disponibilité et réduction des coûts de maintenance. La différence de coûts doit donc être évaluée non seulement en termes financiers, mais également en fonction du temps de gestion et du risque opérationnel associé à chaque technologie.

Utilisation de l'IA dans le projet

Configuration

Contexte:

- Intelligence artificielle utilisé: Chatgpt 5.1 (plus)
- Tous les prompts sont dans un projet Chatgpt nommé "exploration_pocketbase"
- Le projet contient le fichier de l'énoncé du projet 3 ainsi que le mardown de ma planification faite en semaine 1
- Lien du projet: https://chatgpt.com/g/g-p-691b29bf613881919162d7b800a88357-exploration-pocketbase/project

Objectif de cette configuration :

L'objectif est que ChatGPT devienne :

- Un assistant de recherche (résumés, comparaisons, explications techniques),
- Un assistant de développement (génération de code, debugging, optimisation),
- Un assistant de documentation (templates, synthèses, structuration), tout en respectant les exigences du projet et de l'enseignant.

Instruction du projet :

Voici les instructions que mon projet avec tous mes prompts pour le projet 3 avait :

"Rédige tes réponses en tenant compte des consignes du projet 3. Rédige tes réponses en tenant compte de la grille de correction. Réflechis à tes réponses en tenant compte de la planification de la semaine.

Contexte projet 3 :

- Je fais une exploration technique (PocketBase self-hosted vs Supabase non self-hosted).
- Je dois produire du contenu évalué selon une grille (exploration technique, prototype, documentation, journal de bord, utilisation de l'IA).
- Je travaille semaine par semaine selon une planification.

Style attendu:

- Réponds de manière structurée, claire et précise, quittes à manque de créativité
- Utilise des titres, sous-titres et listes pour que je puisse facilement intégrer tes réponses dans mon journal de bord et mon rapport.
- Justifie les choix techniques comme si c'était présenté à mon enseignant.
- Identifie clairement : concepts techniques, démarches, usages de l'IA, limites / risques, alternatives possibles."

Utilisation

Aide à la compréhension technique :

- Comprendre la différence entre posgreSQL et SQLite
- Comprendre en profondeur Supabase et PocketBase
- Comprendre la nuance entre un BaaS hébergé en local et un BaaS hébergé en Cloud
- Comprendre la différence de performance entre Supabase et PocketBase dans le contexte du projet

Génération du code :

- Mise en place d'un backend interchangeable entre PocketBase et Supabase
- Aide dans la structure du projet Svelte
- Aide dans la génération du code pour le CRUD
- Aide dans le debugging avec des modules Svelte
- Mesurer le temps d'execution avec *performance.now()*
- Suggestion de benchmarks (création de la fonction qui test tout les CRUD en même temps_

Aide à la documentation :

- Mise en place d'un prompt Chatgpt (https://chatgpt.com/g/g-p-691b29bf613881919162d7b800a88357-exploration-pocketbase/shared/c/692748ee-93b4-832c-a9fa-44d4957f85a7?owner_user_id=user-JZKWpvthIOnquCb3ituLpCU3) qui permet d'automatiser la génération d'un journal hebdomadaire
- Créer la structure initiale du rapport de synthèse
- Identifier les angles d'analyses pertinents (scalabilité, performance, coûts)
- Valider la justesse et la formulation de mes propos

Critique

L'IA s'est révélée puissante, mais pas dans tous les contextes. Au fil du projet, j'ai constaté que son efficacité dépend surtout de la qualité du prompt et de l'intention derrière chaque demande. Plus un prompt est détaillé et contextualisé, plus l'IA a de chances de produire une réponse pertinente et précise. À l'inverse, un prompt trop vague ou trop ouvert mène souvent à des réponses incomplètes, inventées ou approximatives.

J'ai également remarqué qu'il est généralement préférable de faire une seule demande claire et précise par prompt. Une requête trop large dilue l'attention de l'IA, qui tente alors de « deviner » ce que je veux plutôt que de répondre exactement au besoin. Segmenter mes demandes en petites étapes m'a permis d'obtenir des réponses beaucoup plus fiables, cohérentes et exploitables.

Enfin, même si l'IA est capable de générer du contenu, je l'ai trouvée meilleure pour améliorer que pour créer à partir de zéro. La stratégie la plus efficace a été de commencer par un brouillon puis de lui demander d'améliorer la clarté, la structure ou la formulation. Dans ces situations, l'IA agit comme un assistant éditorial : elle restructure mes idées, propose des pistes et m'aide à atteindre un meilleur résultat sans inventer des informations fausses.

En résumé, l'IA est un outil extrêmement utile lorsque l'on sait comment l'utiliser :

- Fournir du contexte mais éviter de surcharger en détails inutiles,
- Formuler des demandes précises et ciblées,
- Utiliser l'IA comme partenaire d'amélioration plutôt que comme générateur autonome.

Ces apprentissages ont grandement influencé ma manière de travailler tout au long du projet et m'ont permis d'intégrer l'IA de façon plus critique, plus efficace et plus professionnelle.

Reflexion critique

- La technologie est excellente, mais tant que le contexte ne demande pas trop de requetes à la fois
- Sa simplicité est un couteau à double tranchant : améliore la vitesse, la facilité de développement, mais sacrifie certains aspects importants
- Peut notamment être idéale pour des prototypes, des systèmes embarqués, des projets personnels
- Mais n'est pas le choix idéal pour une application web qui demande plusieurs milliers de requêtes en même temps
- Le moteur derrière une technologie BaaS est souvent ce qui dicte les forces et faiblesse

Au terme de cette exploration, j'en viens à reconnaître que PocketBase est réellement excellente, mais uniquement dans un contexte adapté à ses forces. Tant que l'application ne demande pas un volume important de requêtes simultanées, PocketBase offre une expérience remarquable : rapidité, simplicité, déploiement minimal et une courbe d'apprentissage extrêmement douce. Toutefois, cette force devient rapidement une limite dès que le projet commence à prendre de l'ampleur.

Sa simplicité, qui constitue l'un de ses principaux attraits, est en réalité un couteau à double tranchant. D'un côté, elle améliore la vitesse de développement, réduit la complexité et permet de déployer une API et une base de données en quelques minutes. De l'autre, cette même simplicité repose sur des compromis architecturaux importants : SQLite ne gère qu'une seule écriture à la fois, l'absence de scalabilité horizontale limite la croissance, et plusieurs aspects de sécurité ou de performance exigent une vigilance manuelle.

Avec cette perspective, PocketBase apparaît comme un excellent choix pour certains types de projets, notamment :

- les prototypes rapides
- Les projets personnels ou éducatifs
- Les applications embarquées ou à faible trafic
- Les MVP où l'objectif principal est d'itérer vite et à faible coût

En revanche, ce n’est pas le choix optimal pour une application web nécessitant des milliers de requêtes simultanées, des garanties de disponibilité, ou une scalabilité robuste. Dans ces contextes, Supabase (ou toute solution basée sur PostgreSQL) s’impose naturellement : elle est conçue pour absorber une forte charge, gérer des transactions complexes et offrir un environnement cloud entièrement géré.

Ce projet m’a aussi fait réaliser que le moteur sous-jacent d’une solution BaaS dicte en grande partie ses forces et ses faiblesses. SQLite donne à PocketBase sa vitesse et sa simplicité, mais impose aussi ses limites structurelles. À l’inverse, PostgreSQL donne à Supabase une robustesse incomparable, mais au prix d’une plus grande complexité et d’un coût plus élevé. Comprendre ce qui se passe “sous le capot” est donc essentiel pour choisir une technologie adaptée à un contexte donné.

Conclusion

Cette exploration m’a permis de comprendre en profondeur les différences fondamentales entre PocketBase et Supabase, autant sur le plan technique que pratique. PocketBase se démarque par sa simplicité, sa rapidité et son faible coût, mais reste limité dès qu’une application exige scalabilité et charge élevée. Supabase, à l’inverse, offre une robustesse et une flexibilité professionnelles, au prix d’une plus grande complexité et d’un coût supérieur.

Au-delà de la comparaison, ce projet m’a appris à analyser une technologie en fonction de son contexte d’usage, et à utiliser l’IA comme un véritable outil d’appui critique. Cette démarche m’a permis de développer une compréhension plus nuancée du backend et d’améliorer ma capacité à choisir la bonne solution pour un besoin réel.