Cégep du Vieux Montréal

Projet 3

Veille Technologique

**Réalisé par**

Gabriel Veilleux

Arslan Khaoua

Romeo Barraza

# Table des matières

[Table des matières 2](#_Toc197931254)

[Introduction 3](#_Toc197931255)

[Discussion 4](#_Toc197931256)

[Expérience avec les technologies à l’étude 4](#_Toc197931257)

[Pourquoi ces technologies sont intéressantes au sens large 4](#_Toc197931258)

[Interopérabilité 6](#_Toc197931259)

[Veille 7](#_Toc197931260)

[Algorithme Minimax 7](#_Toc197931261)

[Firebase Realtime Database 7](#_Toc197931262)

[Jetpack Compose 7](#_Toc197931263)

[Conclusion 7](#_Toc197931264)

[Impressions 7](#_Toc197931265)

[Générales 7](#_Toc197931266)

[Spécifiques au projet synthèse 7](#_Toc197931267)

[Le futur des technologies étudiées 7](#_Toc197931268)

[Références 8](#_Toc197931269)

[Bibliographie 8](#_Toc197931270)

# Introduction

# Discussion

## Expérience avec les technologies à l’étude

* Algorithme Minimax
* Firebase Realtime Database

Firebase Realtime Database m’a était très utile tout au long de mon projet en tant que base de données. Elle est très facile d’utilisation et possède une vaste documentation ce qui rend sont implémentation ainsi que le débogage très simple. Grace a sa structure JSON et à la synchronisation e temps réel, elle permet un échange fluide et rapide des données entre les utilisateurs.

Cependant, dans le cadre de notre projet, un jeu de shogi joueur contre IA, cette fonctionne de temps réel n’était pas si indispensable que je le pensais. En effet, notre système ne nécessite pas vraiment une synchronisation instantanée des données entre plusieurs clients, ce qui est l’un des principaux avantageux de Firebase Realtime Database. Un model NoSQL plus classique que ce soit les autres model de firebase comme Cloud FireStore ou d’une autre compagnie (MongoDB) aurais peut-être était plus adapter pour les performances ou faciliter

* Jetpack Compose

## Liens présentés dans votre projet de recherche

* Algorithme Minimax
* Firebase Realtime Database

Nous avons modifié notre implémentation de nos bases de donner, nous avons supprimé la tables HistoriqueCoups et l’avons remplacé par un String Json contenant tous les coups des deux joueurs que nous avons ajouté à la table Partie. Nous avons rajouté quelques champs comme le champs partieFini dans la table partie nous permettant de savoir si la partie a était terminer ou non pour pouvoir la continuer plus tard.

* Jetpack Compose

Notre approche à la programmation de l’interface graphique a changé pendant le déroulement du projet 2 de Veille Technologique et nous continuons à l’implémenter sans aucune altération.

## Pourquoi ces technologies sont intéressantes au sens large

* Algorithme Minimax
* Firebase Realtime Database

Selon l’article de Betanet, le Realtime Database est idéal pour des cas comme les chats en direct, les tableaux de bord collaboratifs ou encore les jeux multijoueur. Ces cas s’éloignent de notre jeux joueur contre IA, ce qui montre que le choix technologique bien que fonctionnel aurais pu être affine.

D’un point de vue technique, l’optimisation des performances est cruciale avec Realtime Database. Par exemple, le blog Poespas insiste sur l’importance d’une bonne structuration des données pour éviter les appels redondants ou couteux en bande passante. Ce que nous avons réussi à règle au fur et à mesure de l’avancement de notre projet, permettant à certaines lectures de données d’être simplifiées

* Jetpack Compose

Compose cherche principalement à enlever l’usage de XML. Plutôt que de coder en XML pour faire uniquement des widgets et définir son comportement dans un autre fichier avec Java ou Kotlin, Compose centralise la logique et l’interface graphique sous un même langage de programmation qui est Kotlin. « *Il est beaucoup plus facile de tracer le code, car il est écrit dans le même langage et souvent dans le même fichier. Plus besoin de jongler entre le Kotlin et le XML* (Monzo) ». <https://developer.android.com/develop/ui/compose/why-adopt> – 10/05/25

En plus, cela réduit les étapes pour la création des widgets avec l’utilisation des composants. Il n’est plus nécessaire de définir chaque widget avec XML, les initialiser sur le code Java/Kotlin avec la méthode findViewById et redéfinir leurs comportements, parce que Compose fait la création des widgets en appelant des fonctions modulaires qui rend possible la réutilisation du code. <https://androidknowledge.com/why-jetpack-compose-is-better-than-xml/> - 10/05/2025

Compose devient de plus un plus attirant pour le développement des interfaces graphiques grâce au fait qu’Android est un système d’exploitation qui s’étend à des appareils comme les montres ou les téléviseurs.

<https://github.com/android/wear-os-samples/tree/main/ComposeStarter>

<https://google.github.io/automotive-design-compose/>

<https://developer.android.com/training/wearables/compose>

Compose est très similaire à React Native. Ils implémentent la programmation déclarative, ils utilisent des fonctions pour la création des widgets et ils utilisent des états pour mettre à jour les widgets

<https://medium.com/@vimleshyadavparv/react-vs-jetpack-compose-a-friendly-comparison-of-two-declarative-ui-frameworks-b8b6a94516fa> - 10/05/2025

## Interopérabilité

Enfin, en termes d’interopérabilité, la très grande faciliter d’intégration de Firebase nous a permis d’intégrer d’autre fonctionnalité comme l’authentification et le stockage de fichier en passant par Jetpack Compose, créant ainsi un système cohérent et facile à maintenir. L’engin du jeu (serveur Firebase et algorithme Minimax) a été codé en Java et l’interface graphique (Jetpack Compose) a été codé en Kotlin. Java et Kotlin sont 100% interopérable ce qui a permis une très grande facilité d’intégration des systèmes.

## <https://kotlinlang.org/docs/faq.html>

# Veille

## Algorithme Minimax

## Firebase Realtime Database

Depuis la dernière vielle nous n’avons pas changer ou modifier notre implementation de Firebease Realtime DataBase

## Jetpack Compose

Une nouvelle version de Jetpack Compose (v1.8) est sortie le 25 avril 2025. Une des nouvelles fonctionnalités est l’amélioriation des animations, la facilitation de la saisie automatique des données pour le composable TextField. De plus, le composable Text supporte le formatage HTML, la taille du texte change automatiquement selon la taille du conteneur et il est possible remplacer le texte par un ellipses s’il déborde le conteneur.

Étant donné que l’impact des nouvelles fonctionnalités est léger, il n’y a rien qui affecte drastiquement l’interface graphique du projet.

https://android-developers.googleblog.com/2025/04/whats-new-in-jetpack-compose-april-25.html

# 

# Conclusion

## Impressions

##### Générales

##### Spécifiques au projet synthèse

## Le futur des technologies étudiées

* Algorithme Minimax
* Firebase Realtime Database

Une base de données très solide et sécuritaire, déjà beaucoup utiliser par des grosses entreprises comme Twitch, Duolingo et soutenue par google. Grâce flexibilité elle peut être utiliser que ce soit pour du développement web que Android, rendant la technologie comme une norme à connaitre.

* Jetpack Compose

C’est un logiciel qui continue d’être developpé et promu par Google/Android. Il est évident que Compose continuera à s’étendre à des appareils ayant Android comme système d’exploitation et pourrait être utilisé dans d’autres domaines comme le développement web.

# Références

# Bibliographie

* Source pour firebase :

<https://betanet.net/view-post/exploring-firebase-realtime-database-use>?

<https://blog.poespas.me/posts/2024/05/20/firebase-realtime-database-optimization/>? 3/05/2035

<https://dev.to/vtsen/integrate-firebase-realtime-database-and-user-authentication-into-your-android-app-7fg>?

<https://www.geeksforgeeks.org/data-organization-in-firebase-realtime-database/>?

<https://blog.mindorks.com/firebase-realtime-database-android-tutorial/>?

* Source pour Jetpack Compose :