Cégep du Vieux Montréal

Projet 3

Veille Technologique

**Réalisé par**

Gabriel Veilleux

Arslan Khaoua

Romeo Barraza

# Table des matières

[Table des matières 2](#_Toc198626411)

[Introduction 4](#_Toc198626412)

[Discussion 5](#_Toc198626413)

[Expérience avec les technologies à l’étude 5](#_Toc198626414)

[Algorithme Minimax 5](#_Toc198626415)

[Firebase Realtime Database 5](#_Toc198626416)

[Jetpack Compose 5](#_Toc198626417)

[Liens présentés dans votre projet de recherche 6](#_Toc198626418)

[Algorithme Minimax 6](#_Toc198626419)

[Firebase Realtime Database 6](#_Toc198626420)

[Jetpack Compose 6](#_Toc198626421)

[Pourquoi ces technologies sont intéressantes au sens large? 7](#_Toc198626422)

[Algorithme Minimax 7](#_Toc198626423)

[Firebase Realtime Database 7](#_Toc198626424)

[Jetpack Compose 7](#_Toc198626425)

[Interopérabilité 9](#_Toc198626426)

[https://kotlinlang.org/docs/faq.html 9](#_Toc198626427)

[Veille 10](#_Toc198626428)

[Algorithme Minimax 10](#_Toc198626429)

[Firebase Realtime Database 10](#_Toc198626430)

[Jetpack Compose 10](#_Toc198626431)

[Conclusion 11](#_Toc198626432)

[Impressions 11](#_Toc198626433)

[Générales 11](#_Toc198626434)

[Spécifiques au projet synthèse 11](#_Toc198626435)

[Le futur des technologies étudiées 11](#_Toc198626436)

[Algorithme Minimax 11](#_Toc198626437)

[Firebase Realtime Database 11](#_Toc198626438)

[Jetpack Compose 11](#_Toc198626439)

[Références 12](#_Toc198626440)

[Bibliographie 12](#_Toc198626441)

# Introduction

Pour la réalisation de notre projet, nous avons intégré les technologies : l’algorithme Minimax, pour la programmation de l’intelligence artificielle qui joue contre le joueur, Firebase où on peut sauvegarder des données sur un serveur et Jetpack Compose pour la réalisation de l’interface graphique. Dans ce document, nous exposerons les avantages et les défis pour chacune de ses technologies.

# Discussion

## Expérience avec les technologies à l’étude

### Algorithme Minimax

L’algorithme Minimax était un très bon candidat pour notre joueur « bot » dans notre jeu de Shogi. C’est un algorithme assez intuitif et facile a représenté visuellement. Étant donné que le son procédé soit simple en lui-même et divisé en à peu près 3 étapes principale : la génération d’états, l’évaluation et la recherche fait en sorte qu’il était facile prévoir d’avance dans le développement de notre moteur pour pouvoir l’intégrer. Étant donné que notre moteur soit grandement orienté objets, notre abstraction et encapsulation nous a permis de suivre un « *blueprint* » déjà établit et au besoin, il était facile de créer des classes et méthodes utilitaires. Cependant, cette approche à fait en sorte que pouvoir avoir un Minimax fonctionnel, il a fallu développer jusqu’à quasi-complétion le moteur de Shogi ce qui nous a pris beaucoup de temps. De plus, cette approche nous a aussi forcer a refactoriser certaine partie du code, surtout la logique des pièces.

### Firebase Realtime Database

Firebase Realtime Database nous 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 qu’ont le pensaient. 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**

Lors de la réalisation du design du projet, l’équipe s’attendait à travailler avec un logiciel assez similaire avec ce qui a été enseigné dans le cours de programmation d’applications mobiles, mais comme mentionné dans le document du projet 2, Compose utilise le paradigme de la programmation fonctionnel ce qui a fait changer notre approche et notre structure du projet parce qu’on s’attendait à travailler avec des objets qui représentent des widgets. De plus, il y a d’autres aspects dans la programmation qui sont complètement différents lorsqu’on utilise Compose plutôt que XML, comme l’utilisation d’un seul objet Activity pour toute l’application ou la gestion de la navigation entre les écrans. Finalement cela nous a permis d’explorer d’autres paradigmes de la programmation

## Liens présentés dans votre projet de recherche

### Algorithme Minimax

Notre implémentation de l’algorithme Minimax demeure la même depuis le projet 1 de Veille Technologique.

### 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

L’algorithme Minimax était un choix pertinent pour notre IA de Shogi en raison de sa simplicité d’implémentation, de sa transparence et de sa compatibilité avec une approche orientée objet. Contrairement aux méthodes d’apprentissage automatique, il ne nécessite ni phase d’entraînement ni base de données, ce qui le rend plus accessible et léger pour des projets de taille similaire au nôtre. Il permet une intégration claire d’une fonction d’évaluation adaptée aux spécificités du Shogi et offre également un bon contrôle sur la logique de décision du « bot ». Comparé à l’algorithme ***Monte Carlo Tree Search (MCTS)***, souvent utilisé dans les IA de jeux modernes, Minimax est plus déterministe et prévisible, ce qui facilite le débogage et la validation. *MCTS*, bien que plus puissant dans des environnements à forte complexité comme le Go, repose sur des simulations aléatoires et peut être plus difficile à adapter sans données ou ressources conséquentes. Minimax, enrichi par des optimisations classiques comme l’élagage alpha-bêta, reste ainsi une base solide pour développer une IA stratégique de niveau intermédiaire.

### 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/compdose/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. Finalement, la majorité de nos actions au sein de ces technologies, principalement *FireBase* et *Minimax*, présente une latence dans leur exécution. Cette latence nous a obligés à implémenter plusieurs ***callbacks*** pour gérer les opérations de manière fluide et garantir leur bon enchaînement.

## <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

La réalisation du projet nous a laissé perplexe lorsqu’on constate à quel point le domaine de l’informatique est vaste et cela a éveillé un certain désir à continuer à étudier des algorithmes, paradigmes et logiciels comme ceux que nous avons travaillés avec, surtout qu’une connaissance variée permet de réaliser un meilleur dessin des logiciels.

### Spécifiques au projet synthèse

Malgré les difficultés rencontrées, nous avons réussi à implémenter les technologies (Minimax, Compose et Firebase) à notre projet qui s’approche à ce que nous avons conçu dès le début. Elles se complémentent bien et chacune d’elles offrent des fonctionnalités qui rendent l’application intéressant. Avec plus de temps et avec plus d’expérience pour chacune des technologies étudies, cette application a le potentiel d’atteindre une version plus optimisée avec une intelligence artificielle plus performante.

## Le futur des technologies étudiées

### Algorithme Minimax

Minimax reste un algorithme solide et fiable malgré son âge en raison de sa simplicité, de sa structure claire et de son efficacité dans des jeux à information parfaite. Son principe fondamental, qui consiste à explorer les coups possibles de chaque joueur et à choisir le meilleur, garantit une prise de décision logique et déterministe. Bien que des techniques plus récentes, comme l'apprentissage automatique existent, Minimax demeure pertinent grâce à sa facilité d'implémentation et à ses nombreuses optimisations. Son efficacité reste particulièrement notable dans des situations où l’espace de recherche est limité, et il offre un excellent compromis entre performance et prévisibilité. Un exemple d’utilisation moderne de Minimax peut être trouvé dans les jeux vidéo, où il est encore utilisé dans des jeux de stratégie en temps réel ou de simulation, comme dans certains moteurs de jeux classiques pour des décisions tactiques précises.

### 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 :