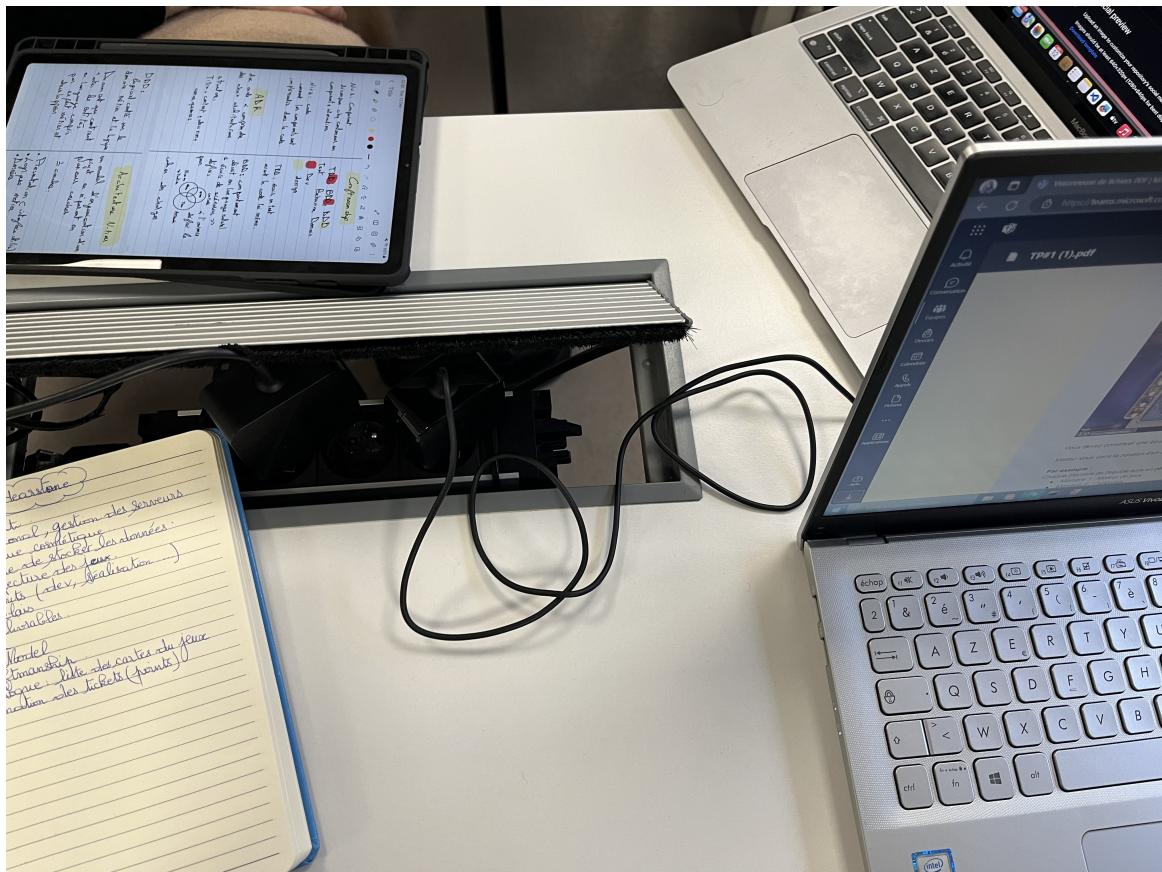


# Hearthstone



Journal de bord de la conception du jeu **Hearthstone Champs de bataille**; un jeu de cartes et de stratégie qui oppose huit joueurs qui dégainent tour à tour leurs cartes. Celui étant le dernier à avoir des points de vie emporte la partie.

- ▼ Etape 1 : Brainstorming en groupe.



▼ Etape 2 : 🔎 Clarification des différentes fonctionnalités à implémenter. (Individuellement)

😊 Kaira : Moteur jeu

☺ Gabrielle : Utilisateurs

📘 Lylia : Catalogue

🛒 Tania : Paiement / Abonnement

▼ Etape 3 : 🏠 Concertation des membres du groupe sur le choix des technos.

### Choix de l'architecture :

#### Microservice :

- Différentes instances à gérer séparément.
- Divers base de données à mettre en place.
- Facilité de maintenance.
- Facilité à remonter les bugs.
- Séparer les différents aspects du jeu.
- Evolution facile.

#### Choix des langages et frameworks

## Moteur de jeu : Unity.

▼ Pourquoi Unity et pas un autre moteur de jeu ?

 Langage utilisé : C#

▼ Pourquoi C# ?

 Système de gestion de base de donnée :

▼ Pourquoi ce choix ?

 [Choix des langages et frameworks](#)

 [Choix de l'architecture](#)

## ▼ Etape 4 : Rédaction de l'ADR

Contexte : Dans le cadre du développement du jeu Hearthstone2, le choix de l'architecture à suivre et des technologies à utiliser est nécessaire.

### Décisions:

- Architecture ⇒ Microservices.
- Moteur de jeu ⇒ Unity.
- Langage de programmation ⇒ C#.
- Gestion des bases de données ⇒ MongoDB, LootLocker.
- Hébergement ⇒ ScalaCube.

## 1- Architecture :

### Microservices :

- Différentes instances à gérer séparément.
- Diverses bases de données à mettre en place.
- Facilité de maintenance.
- Facilité à remonter les bugs.
- Séparer les différents aspects du jeu.
- Evolution facile.

⇒ Statut : Accepté

### Conséquences:

#### Positives:

- Flexibilité.
- Disponibilité : si un souci survient sur l'un des services, l'application sera toujours disponible.
- Scalabilité.

#### Négatives:

- Latence.
- Mise en place fastidieuse.

## 2- Choix du moteur de jeu :

Propositions:

- Unreal Engine
- Unity

Comparaison :

- **Facilité d'utilisation** : Unity est souvent considéré comme plus convivial pour les débutants en raison de son interface intuitive et de sa courbe d'apprentissage moins abrupte par rapport à Unreal Engine. Les développeurs novices peuvent trouver plus facile de commencer à créer des jeux ou des applications avec Unity.
- **Multiplateforme** : possibilité de déployer sur plusieurs plateformes telles que PC, consoles de jeu, appareils mobiles, réalité virtuelle (RV) et réalité augmentée (RA).
- **Graphismes avancés** : prise en charge des graphismes avancés et des effets visuels impressionnantes, ce qui permet aux développeurs de créer des environnements et des personnages attrayants pour leurs jeux ou applications.
- **Vaste écosystème** : une vaste communauté de développeurs qui partagent des ressources, des tutoriels, des plug-ins et des solutions. Cela facilite le processus de développement et permet aux développeurs d'apprendre les uns des autres.
- **Performances** : optimisé pour offrir de bonnes performances, ce qui est crucial pour les jeux et applications interactifs.
- **Évolution constante** : Unity est constamment mis à jour avec de nouvelles fonctionnalités, des améliorations de performances et des correctifs de bugs, ce qui en fait une plateforme évolutive pour les développeurs.
- **Scripting** : Unity utilise le langage de script C# pour la programmation, ce qui est familier à de nombreux développeurs et souvent considéré comme plus accessible que le langage de script propre à Unreal Engine, le Blueprint.
- **Coût** : Comparé à Unreal Engine, bien qu'Unity offre une version gratuite, les coûts de licence et d'abonnement pour des fonctionnalités avancées peuvent être moins élevés que ceux d'Unreal Engine, ce qui peut être un facteur déterminant pour certains développeurs ou petites équipes.

⇒ Statuts :

Unity ⇒ Accepté

Unreal Engine ⇒ Refusé

Conséquences:

Positives:

- Facile à utiliser.
- S'adapte à plusieurs langages de programmation.
- Interopérabilité.

- Tarifs abordable.

Négatives:

- **Graphismes avancés limité.**
- Obligation de commencer un projet from scratch.

### **3- Langage de programmation :**

Propositions:

- C++
- C#
- La langage primaire de Unity est C++ Mais Unity utilise principalement le langage C# comme l'un de ses langages de script et est largement populaire et utilisé dans le développement de jeux.
- Il a aussi une communauté large et très active. C'est une POO facile à apprendre avec ses syntaxes simplifiées.
- Ce qui distingue C# est sa puissante bibliothèque de classes, qui comprend de nombreux **extraits de code pré-écrits** et des **fonctions utilisables** pour construire rapidement des systèmes de jeu complexes.
- En plus des moteurs de jeu et des outils, C# peut également être utilisé pour **les scripts de jeu**.
- Les scripts de jeu sont de petits morceaux de code qui sont exécutés à des points spécifiques pendant l'exécution d'un jeu, comme lorsqu'un personnage joueur subit des dégâts ou lorsqu'un ennemi apparaît.

Ces scripts peuvent être écrits en C# et exécutés par le moteur de jeu, permettant aux développeurs de personnaliser et d'étendre facilement les fonctionnalités de leurs jeux.

- Des nombreux tutoriels de Unity, documentations et ressources sont centrés autour du script en C#. Les devs peuvent utiliser JS pour le script dans Unity mais C# gagne en popularité par rapport à son support, performance et fonctionnalités.

sources:

<https://docs.unity3d.com/2020.1/Documentation/Manual/CreatingAndUsingScripts.html#:~:text=Unity,supports%20the%20C%23%20programming,see%20here%20for%20further%20details.>

c# avec autres langages: <https://www.simplilearn.com/c-sharp-programming-for-beginners-article>

⇒ Statuts :

C# ⇒ Accepté

C++ ⇒ Refusé

Conséquences:

Positives:

- Facile à utiliser.
- Nombreux Documentation et ressources

- Langage principalement utilisé par Unity

Négatives:

- Graphismes avancés limité
- Niveau performance

## 4-Stockage des données :

Propositions:

- Lootlocker
- Firebase
- MongoDB

Comparaison:

**Lootlocker** ( Plateforme qui fournit des services backend pour le dev des jeux )

- **Facilité d'intégration avec Unity** : LootLocker propose un kit de développement Unity (SDK) qui facilite l'intégration de ses fonctionnalités dans les projets Unity. SDK offre des fonctionnalités prêtes à l'emploi, conçues pour fonctionner avec n'importe quel jeu.
- **Gestion de contenu centralisée** : LootLocker permet de prendre en charge tout le contenu du jeu pour chaque plateforme, en un seul endroit. Cela inclut la gestion des cosmétiques, des devises, du contenu généré par l'utilisateur (UGC) et du contenu téléchargeable (DLC).
- **Stockage des données des joueurs** : Toutes les données des joueurs sont stockées au même endroit, ce qui permet d'accéder à leurs profils et listes d'amis sur toutes les plateformes. De plus, la plateforme permet de gérer les rapports, les messages, les remboursements et les cadeaux pour maintenir l'engagement des joueurs.
- **Conformité au RGPD** : LootLocker est conforme au Règlement général sur la protection des données (RGPD) et fournit aux utilisateurs les outils nécessaires pour consulter, gérer et supprimer les données des joueurs.

## Firebase

Avantages :

- Bases de données fiables et étendues .
- Hébergement rapide et sûr.
- Google Analytics : L'intégration avec Google Analytics permet de suivre de manière exhaustive les performances de l'application et l'engagement des utilisateurs.
- Extensibilité : Firebase permet aux développeurs de faire évoluer leurs applications sans stress à mesure que la demande augmente.

## Inconvénients :

- **Coût** : Bien que Firebase propose un niveau gratuit, son modèle de tarification peut devenir coûteux à mesure que le nombre d'utilisateurs et l'utilisation des services augmentent. Il est important de bien comprendre les coûts associés à l'utilisation de chaque service.

- **Flexibilité du modèle de données :** La base de données en temps réel de Firebase est un modèle de données de type JSON, ce qui peut être moins flexible que d'autres modèles de bases de données NoSQL comme MongoDB.
- **Limitations de la requête :** Les requêtes dans Firebase sont limitées par rapport à un système de gestion de base de données plus complet, ce qui peut être un inconvénient pour des opérations complexes.
- **Dépendance envers Google :** Utiliser Firebase signifie dépendre des services de Google, et cela peut être un inconvénient si vous préférez une solution plus indépendante.
- **Personnalisation limitée :** Bien que Firebase propose de nombreuses fonctionnalités, certaines applications très personnalisées ou complexes peuvent nécessiter une personnalisation plus poussée, ce qui peut être limité par rapport à des solutions plus flexibles.

## MongoDB

### Avantages :

- **Évolutivité horizontale :** MongoDB est conçu pour une évolutivité horizontale facile, ce qui signifie qu'il peut s'étendre sur plusieurs serveurs pour gérer des charges de travail plus importantes. Cela permet de faire face à des besoins croissants en termes de volume de données et de trafic.
- **Performance :** MongoDB offre de bonnes performances pour les opérations de lecture et d'écriture, en particulier dans des scénarios où l'accès aux données est souvent en lecture ou en écriture en temps réel, comme dans les applications de jeux multi-joueurs.
- **Indexation riche :** MongoDB prend en charge une variété d'index pour optimiser les requêtes, ce qui peut améliorer significativement les performances de recherche.
- **Langage de requête puissant :** Le langage de requête de MongoDB est puissant et flexible. Il prend en charge des opérations de requête riches qui permettent aux développeurs de récupérer et de manipuler les données de manière efficace.
- **RéPLICATION ET TOLÉRANCE AUX PANNEES :** MongoDB prend en charge la réPLICATION, ce qui permet d'améliorer la disponibilité des données en cas de défaillance d'un serveur. Il offre également des mécanismes de tolérance aux pannes.

### Inconvénients :

- **Consommation de mémoire :** MongoDB peut consommer une quantité significative de mémoire, notamment lorsqu'il est utilisé pour des opérations de lecture et d'écriture intensives. Des configurations appropriées et une surveillance constante peuvent être nécessaires pour gérer la mémoire efficacement.
- **Complexité de la gestion des requêtes :** Bien que MongoDB offre un langage de requête puissant, certaines opérations peuvent être plus complexes que dans des bases de données relationnelles. Les requêtes nécessitant des agrégations avancées ou des jointures peuvent être moins intuitives.
- Pour stocker les états et configuration d'un jeu :

PlayerPrefs (données simple)

JSON (fonctionnalités complexe)

- Pour stocker les données utilisateurs/Payment

⇒ Statuts :

MongoDB ⇒ Accepté

LootLocker ⇒ Accepté

Firebase ⇒ Refusé

## 5- Hébergement :

Propositions:

- **ScalaCube.**
- **Amazon Web Services (AWS)**

**ScalaCube** : est un hébergeur spécialisé dans l'hébergement de serveurs de jeux pour une sélection de jeux populaires tels que Minecraft, Rust, Valheim, CS:GO ..

- Propose des tarifs compétitifs et ses forfaits économiques adaptés à différents budgets, offrant une option plus abordable pour ceux qui cherchent à héberger des serveurs de jeux.

Avantage:

- Spécialisation dans le domaine des jeux et de son orientation vers des forfaits plus économiques.

Inconvénients:

- Limitations de personnalisation avancée.

**Amazon Web Services (AWS)** : AWS offre une large gamme de services cloud, y compris Amazon GameLift pour l'hébergement de serveurs de jeux. Les tarifs varient et peuvent rapidement augmenter en fonction de la taille du serveur, du trafic, des ressources supplémentaires.

Avantage:

- Flexibilité et personnalisation.

Inconvénients:

- Coût.
- Complexité et configuration.

⇒ Statuts :

**ScalaCube** ⇒ Accepté

**AWS** ⇒ Refusé

### ▼ Etape 5 : Craftmanship

⇒ Etape BDD “Behaviour Driven Developpement”



### Fonctionnalité 1: Se connecter “Authentification” :

- Scénario : L'utilisateur se connecte avec les bons identifiants (username et mot de passe).
  - Given : L'utilisateur se rend sur le formulaire de connexion.
  - When : Cliquer sur le bouton « Se connecter ».
  - Then : Accès à la page principale du jeu.
- 

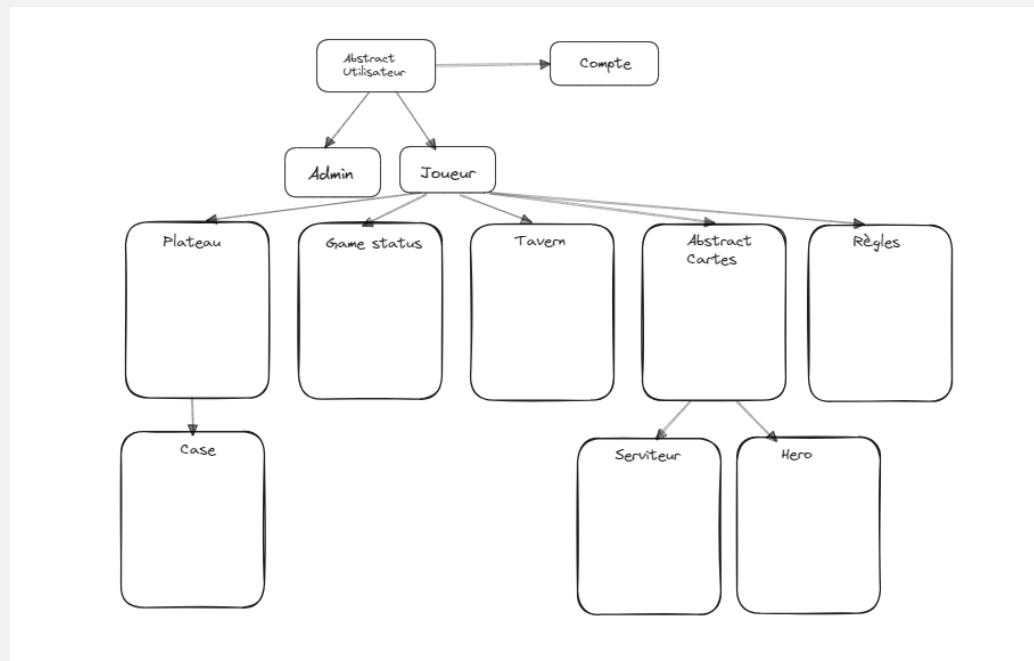
- Scénario : L'utilisateur se connecte avec des identifiants incorrects (username et/ou mot de passe).
  - Given : L'utilisateur se rend sur le formulaire de connexion.
  - When : Cliquer sur le bouton « Se connecter ».
  - Then : Pop-up indiquant que les identifiants sont erronés.
- 

- Scénario : L'utilisateur oublie son mot de passe.
- Given : L'utilisateur se rend sur le formulaire de connexion.
- When : Cliquer sur le lien « Mot de passe oublié ».
- Then : Réception d'un e-mail invitant de réinitialisation de mot de passe.

⇒ Etape DDD “Domain Driven Design”



Il est préférable de ne pas réaliser de DDD dans ce contexte car dans cette application il y a beaucoup de domaines et de composants, les lier entre eux rendrait le modèle trop lourd et difficile à comprendre, on s'éloigne du but du DDD qui est de simplifier la compréhension du projet.

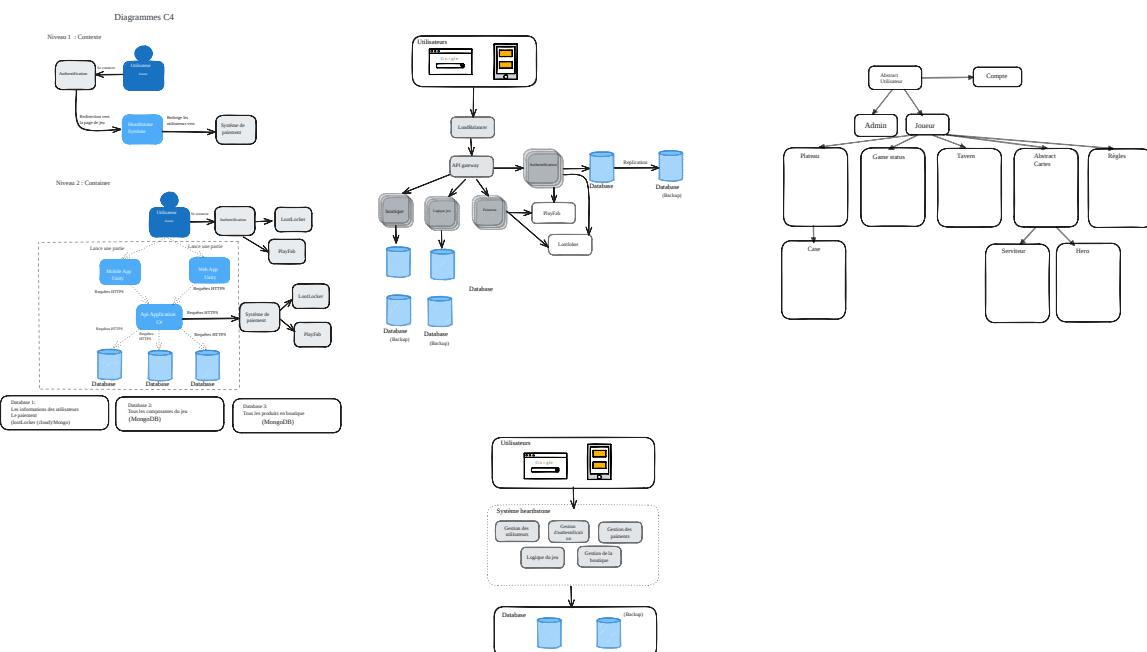
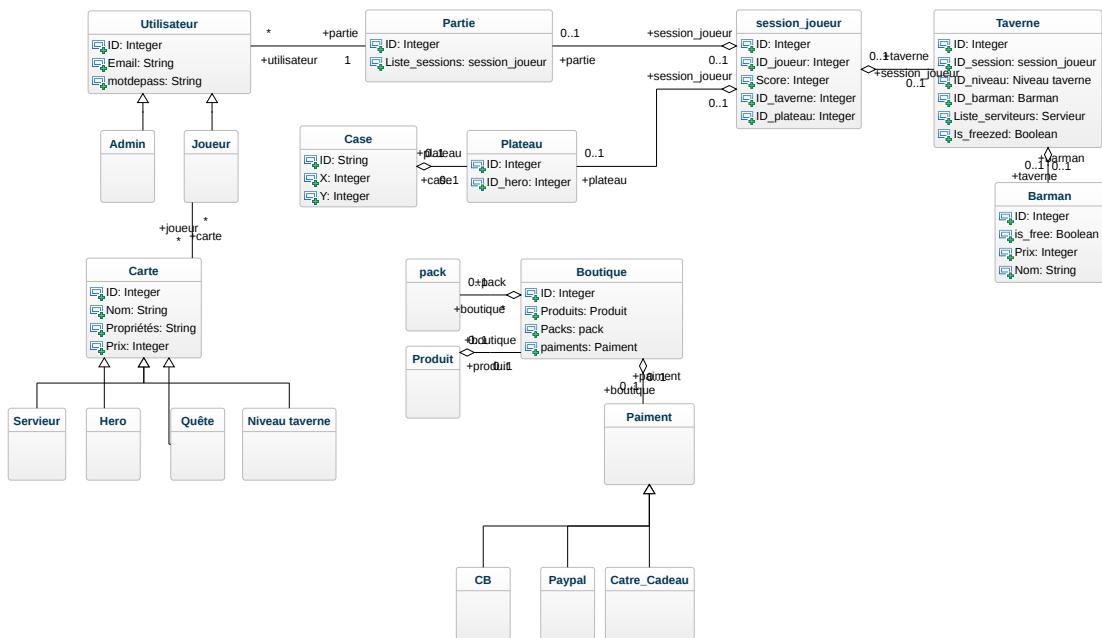


#### ⇒ Etape TDD “Test Driven Developpement”



- Phase pré-code : Initialisation des tests.
- Phase code : Exécution des tests sur le code actuel.
- Phase refacto : Amélioration continue du code en veillant à ce que les tests soient toujours concluants.

#### ▼ Diagrammes



Slide Canva