Fais-moi un dessin

Document d'architecture logicielle

Version 1.7

Historique des révisions

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Auteur** |
| 2020-01-25 | 1.0 | Travail sommaire de l’architecture | Georges Parfait DJIMEFO |
| 2020-01-29 | 1.1 | Rédaction des vues de cas d’utilisation | Georges Parfait DJIMEFO |
| 2020-01-31 | 1.2 | Rédaction des vues logiques | Georges Parfait DJIMEFO |
| 2020-02-03 | 1.3 | Rédaction des vues de processus, de déploiement et de la partie de taille et performance | Georges Parfait DJIMEFO |
| 2020-02-04 | 1.4 | Révision du document entier suite aux remarques des membres de l’équipe notamment sur les vues de cas d’utilisation, logiques et, de déploiement | Georges Parfait DJIMEFO |
| 2020-02-04 | 1.5 | Reformatage des diagrammes de cas d'utilisation et mise à jour des paquetages et du diagramme de déploiement | Ismael Gbian |
| 2020-02-07 | 1.6 | Modification des diagrammes de séquences et correction du français. | Georges Parfait DJIMEFO, Roger Kazma |
| 2020-04-13 | 1.7 | Mise à jour de tout le document. | Ismael Gbian |

Table des matières

[1. Introduction 4](#_Toc37703201)

[2. Objectifs et contraintes architecturaux 4](#_Toc37703202)

[3. Vue des cas d’utilisation 4](#_Toc37703203)

[4. Vue logique 7](#_Toc37703204)

[4.1. Client lourd 11](#_Toc37703205)

[4.2. Client léger 12](#_Toc37703206)

[4.3. Serveur 13](#_Toc37703207)

[4.4. MongoDB Cloud 13](#_Toc37703208)

[4.5. Base de données 13](#_Toc37703209)

[5. Vue des processus 14](#_Toc37703210)

[6. Vue de déploiement 16](#_Toc37703211)

[7. Taille et performance 16](#_Toc37703212)

[7.1 Client léger 17](#_Toc37703213)

[7.2 Client lourd 17](#_Toc37703214)

[7.3 Serveur 17](#_Toc37703215)

[7.4 Base de données 17](#_Toc37703216)

Document d'architecture logicielle

# 1. Introduction

Le présent document décrit la structure de haut niveau d’une application. D’une part, les objectifs et contraintes architecturales y sont définis. D’autre part, les aspects pertinents du modèle de cas d’utilisation, les parties architecturalement significatives du modèle de design, les interactions entre les différents processus significatifs et les configurations de matériel physique sont détaillés. En d’autres termes, la vue des cas d’utilisation, la vue logique, la vue des processus et la vue de déploiement sont développées. Enfin, la description des caractéristiques de taille et de performance pouvant avoir un impact sur l’architecture et le design logiciel est explorée.

# 2. Objectifs et contraintes architecturaux

Étant donné le besoin de communication entre le serveur et un client lourd d’une part, puis le serveur et un client léger d’autre part, l’architecture doit être compatible à un environnement Windows et Android. Dans un souci de réutilisation, le code doit être écrit simplement. Dans la même lancée, il doit être en anglais puisque c’est la langue la plus utilisée. De plus, il doit être “open source” (libre de droits). L’aspect sécurité sera vérifié en veillant à ce que l’architecture assure le cryptage des données personnelles des utilisateurs. Étant donné la grande importance accordée aux volets budget et échéancier, il doit être facilement maintenable. En gros, l’objectif final est de faire une architecture simple qui respecte l’échéancier, les coûts et toutes les contraintes évoquées.

# 3. Vue des cas d’utilisation

Dans cette partie sont présentés les diagrammes de cas d’utilisation les plus pertinents :

### 

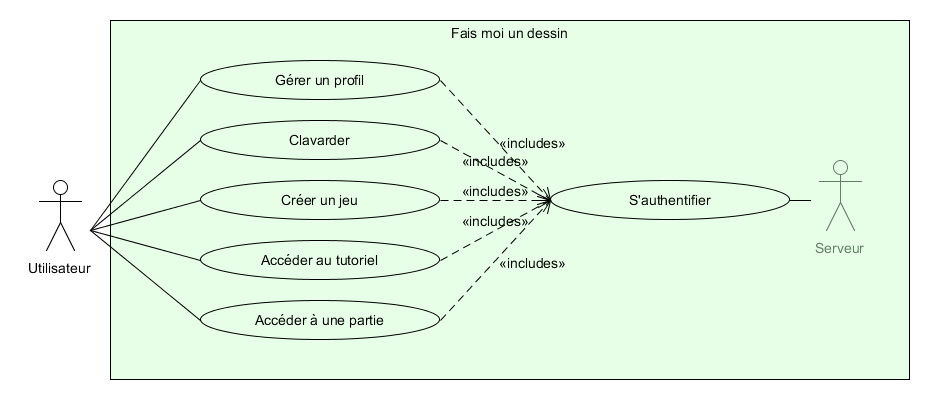


Fig. 1 : Diagramme de cas d’utilisation général

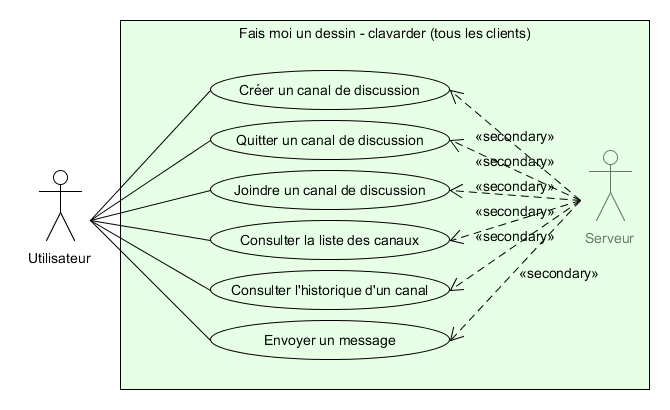


Fig. 2 : Diagramme de cas d'utilisation pour le clavardage

## 

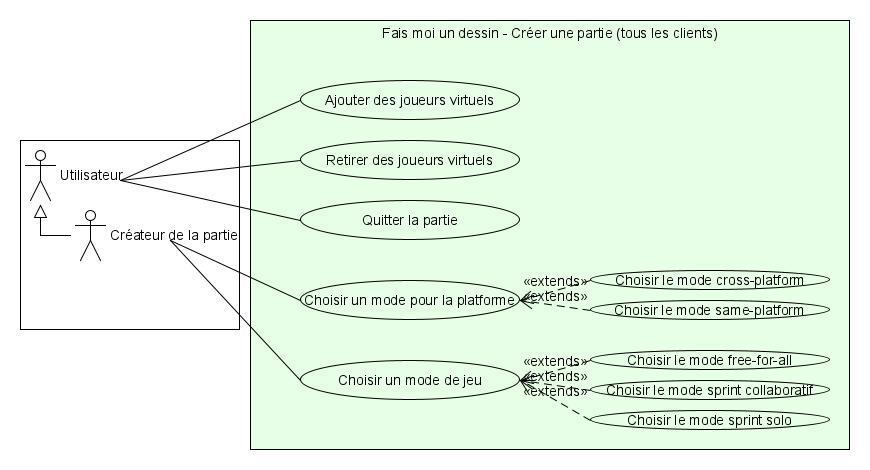


Fig. 3 : Diagramme de cas d’utilisation pour “créer une partie”

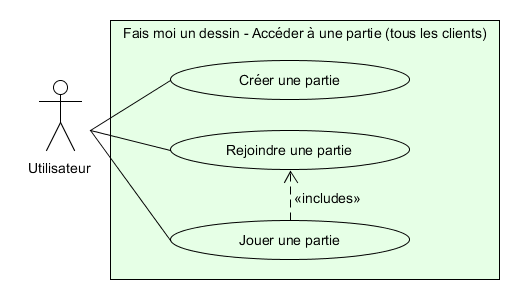


Fig. 4 : Diagramme de cas d’utilisation pour “accéder à une partie”

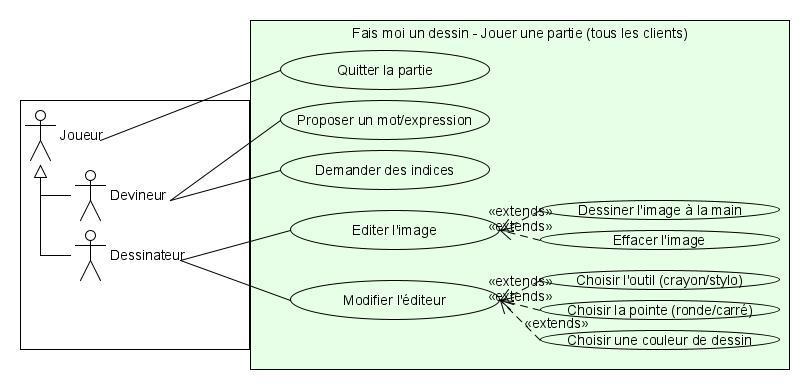


Fig. 5 : Diagramme de cas d’utilisation pour “jouer une partie”

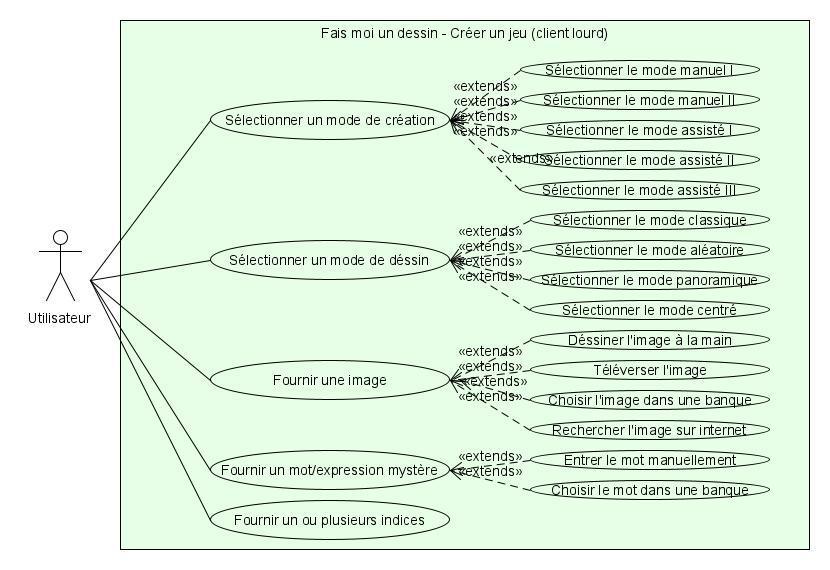


Fig. 6 : Diagramme de cas d’utilisation pour “créer un jeu”

# 

# 4. Vue logique

Les figures 7 et 8 et 9 présentent les diagrammes de paquetages.

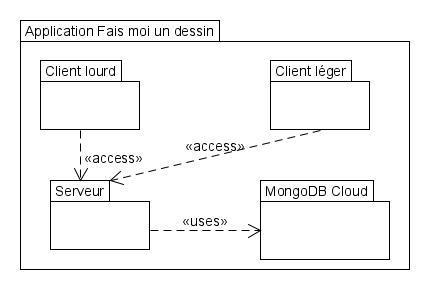


Fig. 7 : Le diagramme de paquetages général

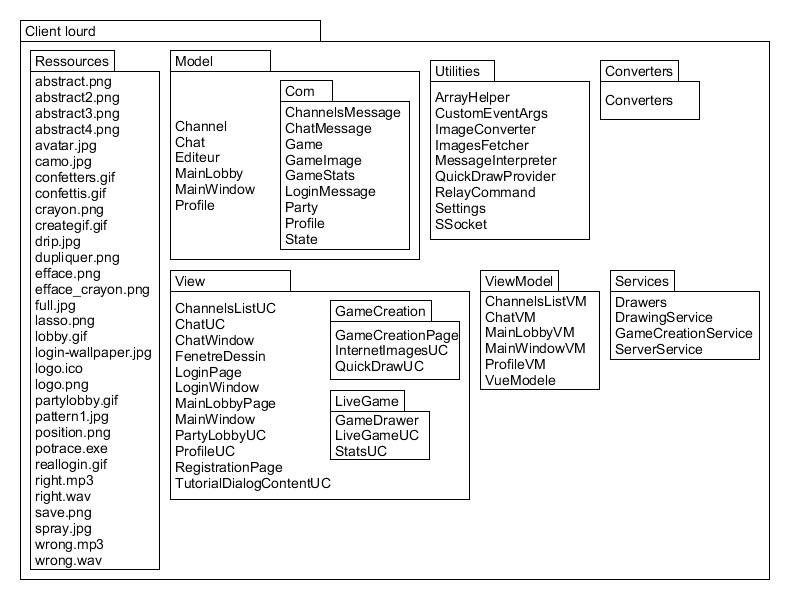


Fig. 8. Diagramme de paquetages du client lourd

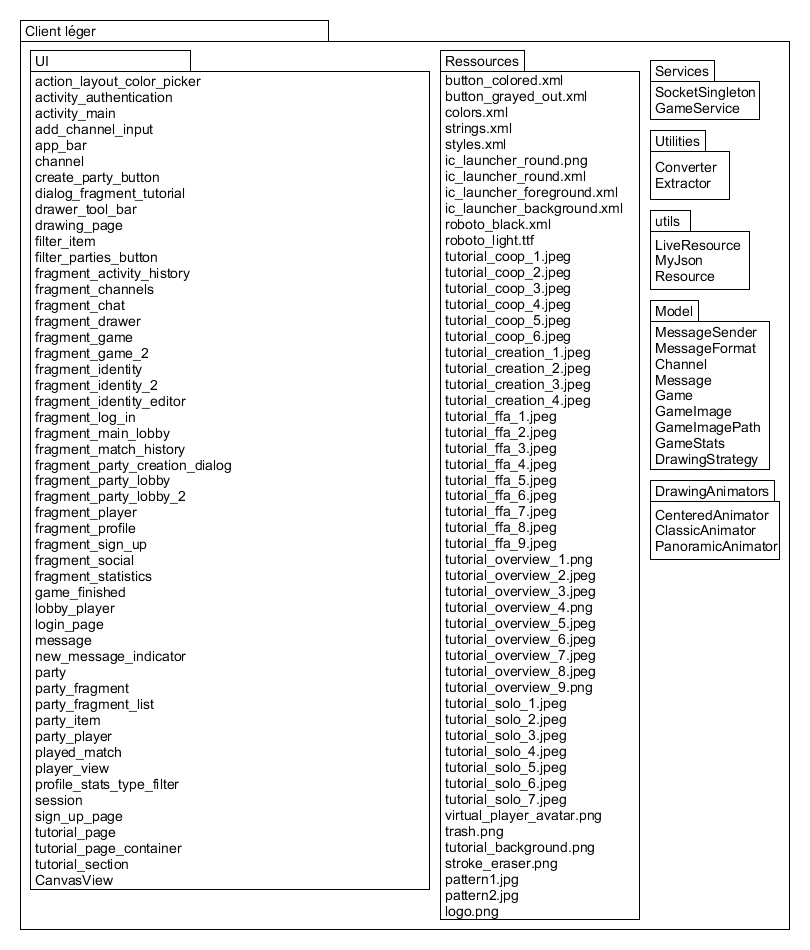


Fig. 9. Diagramme de paquetages détaillé

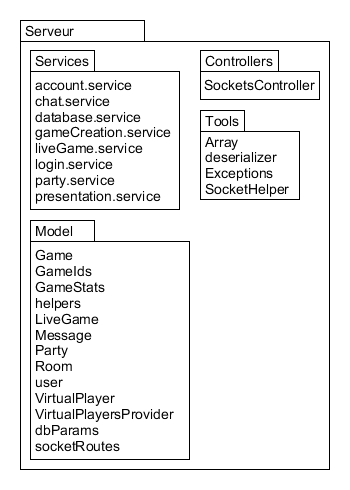


Fig. 10. Diagramme de paquetages détaillé

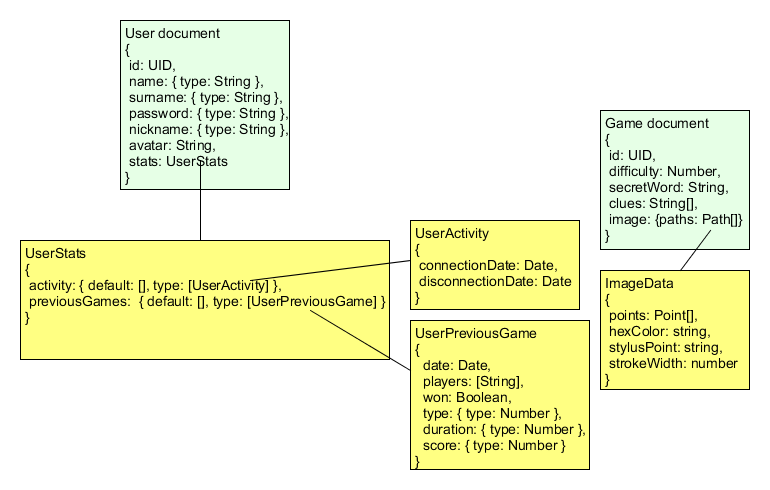


Fig. 11 : Modèle de la base de données non relationnelle

## 4.1. Client lourd

|  |
| --- |
| **Client lourd** |
| Ce paquetage regroupe tout le client lourd. |

|  |
| --- |
| **Ressources** |
| Ce paquetage regroupe toutes les images et les icônes utilisées par le client lourd. |

|  |
| --- |
| **View** |
| Ce paquetage regroupe l’ensemble des fenêtres, pages et sections. Il inclut les vues XAML et leurs fichiers associés. |

|  |
| --- |
| **ViewModel** |
| Ce paquetage regroupe les classes VueModele de l’architecture MVVM. |

|  |
| --- |
| **Model** |
| Ce paquetage regroupe les classes qui s’occupent de la logique de l'application. |

|  |
| --- |
| **Com** |
| Ce paquetage regroupe les classes permettant d'interpréter les messages venants du serveur. |

|  |
| --- |
| **Services** |
| Ce paquetage regroupe les classes qui offrent des services statiques dans toute l'application. |

|  |
| --- |
| **Converters** |
| Ce paquetage regroupe les convertisseurs pour les liaisons de données. |

|  |
| --- |
| **Utilities** |
| Ce paquetage regroupe les paramètres de l'application. |

|  |
| --- |
| **GameCreation** |
| Ce paquetage regroupe toutes les classes utilisées pour la création des jeux. |

|  |
| --- |
| **LiveGame** |
| Ce paquetage regroupe toutes les classes servant à afficher les jeux durant une partie. |

## 

## 4.2. Client léger

|  |
| --- |
| **Ressources** |
| Ce paquetage regroupe toutes les images et les icônes utilisées par le client léger. |

|  |
| --- |
| **UI** |
| Ce paquetage regroupe l'ensemble des vues et activités de l'application. Cela inclut tous les fichiers XML associés. |

|  |
| --- |
| **Model** |
| Ce paquetage regroupe les classes qui s’occupent de la logique de l'application. |

|  |
| --- |
| **Services** |
| Ce paquetage regroupe les classes qui offrent des services statiques dans toute l'application. |

|  |
| --- |
| **Utilities** |
| Ce paquetage regroupe toutes les classes utilitaires de l'application. |

|  |
| --- |
| **utils** |
| Ce paquetage regroupe les classes utilisées pour extraire les ressources de l'application. |

|  |
| --- |
| **DrawingAnimators** |
| Ce paquetage regroupe les classes utilisées pour effectuer les animations (dessins réalisés par un joueur virtuel). |

## 4.3. Serveur

|  |
| --- |
| **Controllers** |
| Ce paquetage regroupe les contrôleurs du serveur. Ceux-ci sont chargés d'assigner chaque requête des clients au service adéquat. |

|  |
| --- |
| **Tools** |
| Ce paquetage regroupe les classes et fichiers servant d'outils dans tout le serveur. |

|  |
| --- |
| **Model** |
| Ce paquetage regroupe les classes qui s’occupent de la logique du serveur. |

|  |
| --- |
| **Services** |
| Ce paquetage regroupe les classes qui offrent des services dans le serveur. |

## 4.4. MongoDB Cloud

|  |
| --- |
| **MongoCloud** |
| Ce paquetage regroupe les classes responsables de la base de données. |

## 4.5. Base de données

|  |
| --- |
| **User document** |
| Ce paquetage regroupe toutes les données concernant les utilisateurs enregistrés dans l'application. |

|  |
| --- |
| **Game document** |
| Ce paquetage regroupe toutes les données concernant les jeux sauvegardés dans l'application. |

# 5. Vue des processus

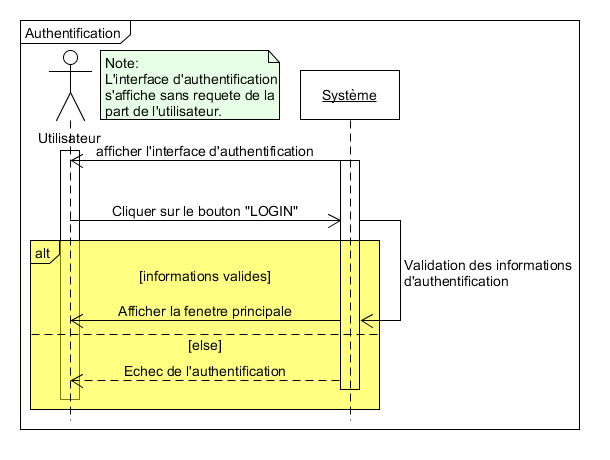


Fig. 12. Diagramme de séquences pour l’authentification

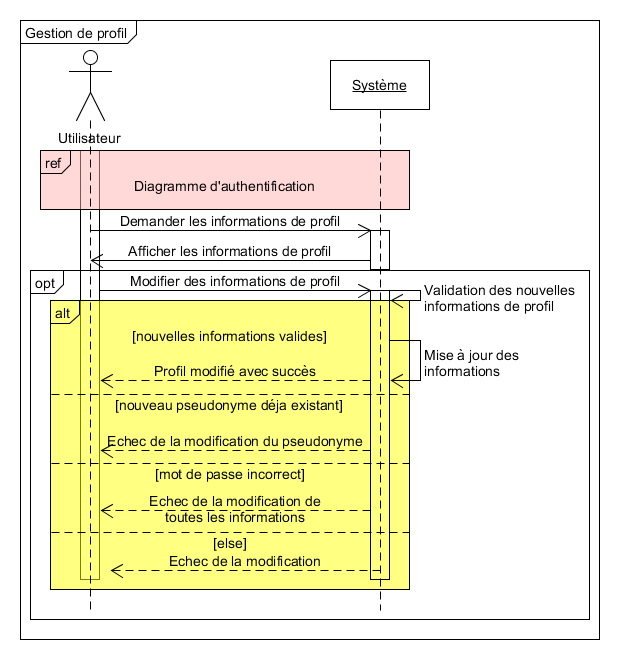


Fig. 13. Diagramme de séquences pour la gestion de profil

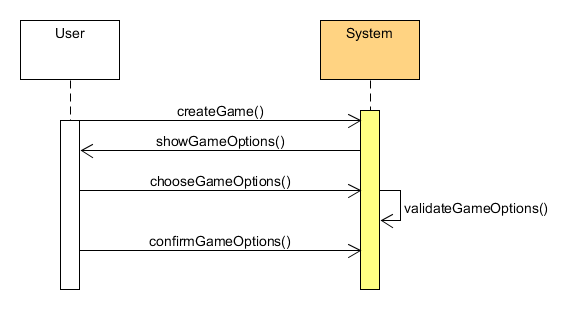


Fig. 14. Diagramme de séquences pour la création d'un jeu

# 6. Vue de déploiement

Le diagramme de déploiement a été utilisé dans ce contexte de vue de déploiement (fig. 13). Les nœuds et leurs interconnexions y sont représentés.

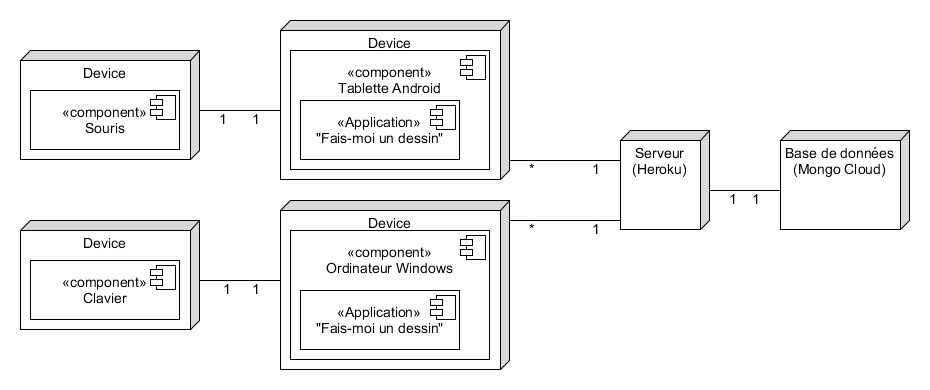


Fig. 15. Diagramme de déploiement

# 7. Taille et performance

Les contraintes de taille et de performance mentionnées dans le SRS sur les paramètres de mémoire ont été prises en compte dans l’architecture développée dans ce document.

Les critères ergonomiques développés par Scapin et Bastien (Réf. <https://www.ergoweb.ca/criteres/>) ont été suivis pour assurer une expérience utilisateur des plus agréables.

## 7.1 Client léger

Les paramètres physiques susceptibles d’entraîner la lenteur tels que la mémoire ont fait l’objet de vérifications. Les spécificités de la tablette Galaxy A (cf. <https://www.samsung.com/ca/tablets/galaxy-tab-a-2019-101/SM-T510NZKAXAC/>) rassurent que les caractéristiques de l’application telles que déjà évoquées permettront qu’elle tourne de façon optimale.

## 7.2 Client lourd

L’application roule sur des ordinateurs Windows qui jouissent de plus de 1 Go de mémoire vive et d’espace disque de plus de 100 Mo. L’application utilise beaucoup moins de ressources que les précédentes (moins de 1 Go de mémoire vive et 50 Mo d’espace). Les facteurs de performances ont donc été validés dans la mise en œuvre de l’architecture.

## 7.3 Serveur

La plateforme Heroku a été désignée pour supporter le serveur. Avec 512 Mo de mémoire vive disponibles pour l'option gratuite, le serveur pourra supporter les utilisateurs prévus sans une dégradation notoire de ses performances. Il sera cependant crucial de surveiller la consommation des ressources du serveur afin de ne pas excéder les limites du plan actuel. Les données des utilisateurs ainsi que les images des avatars seront sauvegardées dans une base de données utilisant MongoDB. Grâce aux performances optimales de ce système, le serveur et les clients pourront presque instantanément obtenir les informations nécessaires au bon déroulement de chaque partie.

## 7.4 Base de données

La base de données est fournie par MongoDB. Avec 512 Mb d'espace disponible ainsi que jusqu'à 500 connexions simultanées autorisées, elle est amplement convenable aux besoins de l'application.