Équipe 203

PolyDraw
Spécifications des requis du système (SRS)

Version 1.2

Historique des révisions

Date	Version	Description	Auteur
2020-01-26	1.0	Version pour la remise en LOG3000	Allan Beddouk Cédric Tessier Samuel Saito-Gagné Martin Pouliot Pascal-Alexandre Morel
2020-02-05	1.1	Ajustements sur les exigences non fonctionnelles suite au retour de la correction de LOG3000	Martin Pouliot
2020-02-07	1.2	Correction de fautes de français	Pascal-Alexandre Morel

Table des matières

Spécifications des requis du système (SRS)

1. Introduction

1.1. But

Le SRS décrit le comportement externe d'une application. Il décrit aussi les exigences non fonctionnelles, les contraintes de conception, ainsi que les autres facteurs nécessaires à la description complète des exigences du logiciel à développer.

1.2. Définitions, acronymes et abréviations

Afin de simplifier et normaliser ce document, voici une brève présentation des abréviations/synonymes que nous utiliserons tout au long du projet :

- Pour parler du logiciel en général, nous avons convenu sur un nom pour le désigner : PolyDraw. Ainsi, lorsqu'on parle de PolyDraw, on parle du projet dans sa globalité.
- Un « jeu » correspond à une association d'une image et d'un mot.
- Une « partie » correspond à un ensemble de jeu.
- Un « joueur humain » correspond à un étudiant en génie logiciel ou des ingénieurs en logiciel.
- Un « IDE » est un environnement de développement intégré. Il inclut donc différents outils pour le développement.
- « WPF » est un cadriciel utilisé pour la conception d'interfaces sur Windows.
- « TCP » est un protocole utilisé pour faire la gestion de session sur le réseau.
- « HTTP » est un protocole utilisé pour obtenir des données sur un réseau depuis un serveur distant.
- « SQL » est un langage utilisé pour faire des requêtes à un SGBD.
- « SGBD » est un système de gestion de base de données.
- « LTS » signifie que la version est supportée pendant une longue période.

1.3. Vue d'ensemble du document

Au sein du document, nous retrouverons plusieurs parties distinctes dédiées à décrire, en détail, les caractéristiques du logiciel. Chaque partie est subdivisée en différentes sous-parties permettant de mieux séparer et organiser nos idées.

Comme second point on peut retrouver une description globale du logiciel. Dans cette dernière on peut retrouver, entre autres, les différentes interfaces supportées par le logiciel ainsi que les outils utilisés dans notre code. Pour les autres sous parties, on retrouve également les contraintes liées au logiciel ainsi que les hypothèses et dépendances que nous avons émises en conséquence.

Par la suite, on retrouve une partie concernant les exigences fonctionnelles du logiciel. Elle va permettre de présenter les fonctionnalités que nous avons décidé d'implémenter.

Enfin, la dernière partie porte sur les exigences non fonctionnelles du logiciel. En fonction des exigences fonctionnelles définies dans le point précédent, ces exigences vont varier. Cette partie va donc plus être orientée sur les caractéristiques techniques du logiciel comme la fiabilité ou encore la performance.

2. Description globale

PolyDraw est un logiciel de dessin qui peut être joué sur mobile ou sur ordinateur. Il peut être joué avec d'autres

personnes ou avec des joueurs virtuels. Le déroulement d'une partie permet à un usager de dessiner une image alors que les autres tentent de deviner le mot associé à l'image dessinée pour accumuler des points. L'application permet également aux utilisateurs d'interagir en leur fournissant des canaux de discussion. Pour chaque utilisateur, des données seront sauvegardées en lien avec leurs performances de jeu (p. ex. le nombre de parties gagnées).

2.1. Caractéristiques des usagers

Les utilisateurs sont des étudiants en génie logiciel et/ou des ingénieurs en logiciel. Il est donc raisonnable d'assumer que les utilisateurs ont une bonne expérience avec l'informatique. Ces utilisateurs seront capables de comprendre rapidement l'interface et ont déjà une expérience avec des applications similaires. Ils sont assez techniques pour rechercher de l'information et soumettre un bogue dans le cas où celui-ci arriverait.

2.2. Interfaces

2.2.1. Interfaces usagers

Le client lourd aura une interface faite à l'aide du cadriciel .NET Framework et WPF alors que le client léger aura une interface construite à l'aide de l'IDE Android Studio et du langage Kotlin. Le serveur est développé en Go. Les clients communiquent avec le serveur à l'aide d'un api REST ainsi qu'un socket TCP pour la partie temps réel.

2.2.2. Interfaces matérielles

Pour le client lourd, l'application devra supporter les moniteurs de différentes résolutions, les claviers, la souris et les pavés tactiles. Pour une expérience-utilisateur complète, les haut-parleurs de l'ordinateur pourraient émettre des sons au cours d'une partie, mais l'utilisateur peut choisir de mettre le jeu en sourdine en réduisant le son de l'ordinateur.

Pour le client léger, il sera nécessaire que l'application soit supportée par une tablette Android de type Galaxy Tab A. Les haut-parleurs de la tablette pourraient être utilisés pour émettre des sons au courant d'une partie, mais l'utilisateur peut choisir de mettre le jeu en sourdine en réduisant le son de la tablette. L'écran tactile de l'appareil sera utilisé pour effectuer les différentes opérations sur l'application.

2.2.3. Interfaces logicielles

Plusieurs interfaces logicielles seront utilisées. Elles sont divisées en trois catégories, soient les systèmes d'exploitation, les cadriciels et les librairies. Pour mieux les représenter, des tableaux prévus à cet effet ont été conçus.

Tableau 1: Présentation des interfaces logicielles du serveur

Type d'interface logicielle	**Nom**	**Utilité**
Système d'exploitation	Ubuntu 18.04	Système d'exploitation sur lequel le serveur sera développé.
Système d'exploitation	Alpine Linux	Utilisé pour le système d'exploitation dans le conteneur Docker du serveur.
Cadriciel	Docker	Utilisé pour le déploiement du serveur.
Langage de programmation	Go	Langage utilisé pour développer le serveur.
Logiciel	Potrace	Logiciel utilisé pour le traçage des bitmaps en SVG.
Librairie	Gorilla Mux	Gère les requêtes HTTP du côté du serveur.
Librairie	Gorm	Permets de se connecter à une base de données SQL et d'effectuer des requêtes.
Librairie	MessagePack	Permets de sérialiser et désérialiser des objets sous le format MessagePack en Go pour les envoyer et les recevoir des clients.
Librairie	Viper	Permets de charger en mémoire les fichiers de configurations et mettre des défauts pour ces fichiers.

Tableau 2: Présentation des interfaces logicielles du client lourd

Type d'interface logicielle	**Nom**	**Utilité**
Système d'exploitation	Windows 10	Permettra d'utiliser et de développer l'application ordinateur (client le lourd).
Cadriciel	.NET Framework v4.7.2	Offre une gamme de librairies facilitant le développement de l'application Desktop.
Cadriciel	Material Design	Facilite la construction de l'interface du client lourd et léger en offrant des outils visuels.
Langage de programmation	C#	Langage utilisé pour développer l'application avec le cadriciel .NET.
Librairie	RestSharp	Permets d'envoyer des requêtes HTTP au serveur à partir du client lourd.
Librairie	MessagePack	Permets de sérialiser et désérialiser des objets en format binaire pour diminuer le délai de communication entre le serveur et les clients.
Librairie	Gong-wpf-dragdrop	Permets de glisser et de déposer une fenêtre (p. ex. glisser et déposer la fenêtre de canal de discussion).

Tableau 3: Présentation des interfaces logicielles du client léger

Type d'interface logicielle	**Nom**	**Utilité**
Système d'exploitation	Android 9	Permettra de rouler l'application mobile.
Cadriciel	Android SDK 28	Permets de développer des applications pour des systèmes Android. Facilite le développement et offre plusieurs librairies pour créer l'application.
Langage de programmation	Kotlin	Langage utilisé pour développer l'application avec le cadriciel Android.
Librairie	Retrofit	Permets d'envoyer des requêtes HTTP au serveur à partir du client léger.
Librairie	MoshiPack	Permets de sérialiser et désérialiser des objets sous le format MessagePack en Kotlin pour les envoyer et les recevoir du serveur.

2.2.4. Interfaces de communication

Pour les communications ne nécessitant pas un temps de réponse instantané, des communications HTTP à l'aide de REST ont été utilisées. Pour les communications en temps réel, des sockets ont été utilisés. Pour ces sockets, le protocole TCP sera utilisé. Les clients enverront des messages à un serveur hébergé à Polytechnique Montréal. Ce serveur a une connexion internet gigabit et elle a 2 liens Ethernet. Le débit de la connexion est symétrique. Pour lancer des machines virtuelles, le serveur utilise ESXi. C'est avec cette technologie qu'une machine virtuelle Ubuntu Server 18.04 LTS avec 4 GB et 30 GB de SSD est lancée. Le serveur en tant que tel a 192 GB de RAM, 40 coeurs, 1 TB de SSD et 10 TB de HDD. Il utilisera également pfSense comme routeur virtuel.

Le client léger et le client lourd peuvent communiquer au serveur grâce à un lien Ethernet ou à la connexion Wifi du réseau de Polytechnique Montréal.

Pour la communication client-serveur, les structures de données JSON seront sérialisées sous un format binaire à l'aide de MessagePack avant d'être envoyées sur le réseau. Ce format binaire sérialisé est plus léger que JSON. L'application permettra aussi de jouer avec d'autres joueurs à l'aide d'internet.

2.3. Contraintes générales

Deux ports du serveur devront être disponibles et configurables en tout temps pour d'abord recevoir les requêtes REST et pour ensuite recevoir les échanges par sockets. Le serveur devra également être capable de soutenir la connexion simultanée de 8 joueurs sans perte de performance. De plus, le client lourd devra être utilisé sur un système d'exploitation Windows 10. Le client léger sera utilisé sur une Galaxy Tab A et sur le système d'exploitation Android 9. Pour la persistance, des images Docker du serveur sont utilisées pour conserver le même environnement lors de l'exécution du serveur sur n'importe quelle structure. Ensuite, puisque la tablette n'a que 2 GO de RAM, il faudra optimiser l'application pour qu'elle fonctionne bien dans une partie de 8. Pour l'application Android, l'application devra être performante sur un réseau de 4G. Du côté de l'application sur ordinateur, le logiciel devra être performant sur un réseau Wifi. Quant à la taille des applications Android et ordinateur, celles-ci ne devront pas dépasser 200 MB. Le serveur devra aussi fonctionner avec une mémoire vive de moins de 512 MB puisqu'il ne fera pas grand-chose.

Comme autres contraintes, chaque membre de l'équipe se verra assigner des tâches qu'il devra finir avant les dates limites précisées dans l'échéancier. De même, l'équipe devra fournir un prototype et différents artefacts avant le 7 février pour répondre à l'appel d'offres. Ensuite, l'interface utilisateur pourra être utilisée en français et en anglais. Pour ce qui en est de la base de données, l'accès aux informations des utilisateurs devra être restreint aux gestionnaires de la base de données. Ces informations doivent rester confidentielles et elles ne peuvent pas être vendues.

Quant à la qualité du jeu multijoueur, les utilisateurs qui jouent ensemble devront recevoir les différentes informations provenant du serveur en même temps, de manière à ce qu'aucun utilisateur ne soit avantagé (p. ex: il ne faut pas qu'un joueur reçoive un indice avant un autre).

2.4. Hypothèses et dépendances

Comme hypothèses générales, il faudra que les usagers aient une connexion internet adéquate et que leur ordinateur soit moindrement performant. Par ordinateur moindrement performant, on parle d'un ordinateur pouvant jouer à des jeux sur navigateur web sans problèmes de latence. De plus, le serveur sera presque toujours disponible. Il n'y aura pas alors de problèmes de déconnexion avec les utilisateurs. Les clients devront toujours afficher des messages

d’erreurs dans le cas où un problème de connexion survient.

3. Exigences fonctionnelles

3.1 Authentification [Essentielles]

3.1.1. Client lourd et léger

3.1.1.1. Le système doit présenter à l’utilisateur la page de connexion à l’ouverture de l’application.
3.1.1.2. Le système doit permettre à l’utilisateur de se connecter.
3.1.1.3. Le système doit permettre à l’utilisateur de se créer un compte.
3.1.1.4. Le système doit s’assurer que le nom de compte de l’utilisateur doit être composé de 4 à 12 caractères alphanumériques.
3.1.1.5. Le système doit s’assurer que le mot de passe de l’utilisateur doit être composé de 8 à 64 caractères.
3.1.1.6. Le système doit avertir l’utilisateur d’un critère non respecté.
3.1.1.7. Le système doit vérifier l’unicité du nom d’utilisateur.
3.1.1.8. Le système doit avertir l’utilisateur après une tentative de connexion échouée.
3.1.1.9. Le système doit empêcher l’utilisateur de se connecter si sa session est déjà ouverte.
3.1.1.10. Le système doit informer l’utilisateur d’une session déjà ouverte.
3.1.1.11. Le système doit accepter les caractères spéciaux pour le mot de passe de l’utilisateur.

3.2. Lobby de jeu [Essentielles]

3.2.1. Client lourd et léger

3.2.1.1. Le système doit présenter à l’utilisateur les parties en attente de joueurs.
3.2.1.2. Le système doit permettre à l’utilisateur de rejoindre une partie en attente de joueurs.
3.2.1.3. Le système doit permettre à l’utilisateur de consulter la liste des joueurs y participant pour chaque partie listée.
3.2.1.4. Le système doit permettre à l’utilisateur de consulter le mode de jeu pour chaque partie listée.
3.2.1.5. Le système doit permettre à l’utilisateur de créer une partie.
3.2.1.6. Le système doit permettre à l’utilisateur de donner un titre à la partie pour la création d’une partie.
3.2.1.7. Le système doit permettre à l’utilisateur de choisir le mode de jeu mêlée générale pour la création d’une partie.
3.2.1.8. Le système doit permettre à l’utilisateur de choisir le nombre de tours de jeu avant la fin de la partie pour la création d’une partie.
3.2.1.9. Le système doit permettre à l’utilisateur de choisir le nombre maximal de joueurs pour la création d’une partie..
3.2.1.10. Le système doit permettre à l’utilisateur ayant créé une partie (l’hôte) de la débuter dès qu’un autre joueur humain rejoint la partie en attente.
3.2.1.11. Le système doit permettre à l’hôte de retirer un joueur de la partie en attente.
3.2.1.12. Le système doit permettre à l’hôte d’ajouter un joueur virtuel à la partie en attente.

3.3. Lobby de jeu [Souhaitables]

3.3.1. Client lourd et léger

3.3.1.1. Le système doit permettre à l’utilisateur de choisir le mode de jeu sprint solo pour la création d’une partie.
3.3.1.2. Le système doit permettre à l’utilisateur de choisir le mode de jeu sprint coopératif pour la création d’une partie.
3.3.1.3. Le système doit permettre à l’utilisateur de choisir la taille des équipes pour la création d’une partie en mode sprint coopératif.
3.3.1.4. Le système doit restreindre la taille des équipes entre 2 et 4 joueurs pour la création d’une partie en mode sprint coopératif.

3.4. **Profil utilisateur et historique [Essentielles]**

3.4.1. Client lourd et léger

3.4.1.1. Le système doit permettre à l'utilisateur de visualiser les informations publiques de son compte.
Les informations publiques sont: le pseudo et l'avatar.

3.4.1.2. Le système doit permettre à l'utilisateur de visualiser les informations privées de son compte.
Les informations privées sont : le nom, le prénom, le mot de passe.

3.4.1.3. Le système doit permettre à l'utilisateur de visualiser les statistiques sur son utilisation de l'application.
Les statistiques sont: le nombre de parties jouées, le pourcentage de victoires, le temps moyen d'une partie et le temps total passé à jouer.

3.4.1.4. Le système doit permettre à l'utilisateur de visualiser l'historique de ses dernières connexions.
L'historique doit permettre de visualiser les heures et dates des dernières connexions.

3.4.1.5. Le système doit permettre à l'utilisateur de visualiser l'historique de ses parties
L'histoire comprend: date, heure, liste des joueurs, mode de jeu et résultat.

3.4.2. Client léger

3.4.2.1. Le système doit permettre à l'utilisateur de changer le thème de couleurs de l'interface parmi 4 choix.

3.5. Construction d'un jeu [Essentielles]

3.5.1. Client lourd

3.5.1.1. Le système doit permettre à l'utilisateur d'accéder à l'interface de construction d'un jeu à partir de l'accueil.

3.5.1.2. Le système doit permettre à l'utilisateur de choisir la langue (français ou anglais) du jeu.

3.5.1.3. Le système doit sélectionner la langue de l'utilisateur par défaut pour la création d'un jeu.

3.5.1.4. Le système doit permettre à l'utilisateur de soumettre le jeu si le champ du mot et les champs des indices sont correctement remplis.
Les critères de ces champs sont définis dans les exigences 3.5.1.5 à 3.5.1.13

3.5.1.5. Le système doit permettre à l'utilisateur de saisir un mot à ajouter au jeu.

3.5.1.6. Le système doit s'assurer que le mot ne contienne que des lettres.

3.5.1.7. Le système doit assurer la validité du mot en le cherchant dans un dictionnaire anglais ou français.

3.5.1.8. Le système doit empêcher l'utilisateur d'ajouter un mot vulgaire en cherchant dans une liste de mots vulgaires.

3.5.1.9. Le système doit assurer l'unicité d'un mot.

3.5.1.10. Le système doit permettre à l'utilisateur de remplir un champ afin d'ajouter un indice pour le jeu.

3.5.1.11. Le système doit s'assurer que l'utilisateur soumet au moins un indice pour le jeu.

3.5.1.12. Le système doit s'assurer qu'un indice n'est pas vide.

3.5.1.13. Le système doit s'assurer qu'un indice soit de moins de 100 caractères

3.5.1.14. Le système doit s'assurer qu'un indice ne contienne pas le mot à deviner.

3.5.1.15. Le système doit permettre à l'utilisateur de fournir des images de type BMP, JPG ou PNG.

3.5.1.16. Le système doit convertir automatiquement l'image en format SVG.

3.5.1.17. Le système doit permettre à l'utilisateur de dessiner l'image dans une zone de dessin.

3.5.1.18. Le système doit permettre à l'utilisateur de choisir la difficulté de son jeu.
Les difficultés sont: facile, moyen ou difficile.

3.5.1.19. Le système doit permettre à l'utilisateur de choisir le mode de dessin de jeu classique.

3.5.1.20. Le système doit permettre à l'utilisateur de choisir le mode de dessin de jeu aléatoire.

3.5.1.21. Le système doit permettre à l'utilisateur de choisir le mode de dessin de jeu panoramique.
3.5.1.22. Le système doit permettre à l'utilisateur de choisir le mode de dessin de jeu centré.
3.5.1.23. Le système doit permettre à l'utilisateur de visualiser un aperçu de l'image dessinée avec le mode de dessin sélectionné.

3.6. **Construction d'un jeu [Souhaitables]**

3.6.1. Client lourd

3.6.1.1. Le système doit proposer à l'utilisateur une série d'images selon le mot entré.

3.6.2. Client léger

3.6.2.1. Le système doit permettre à l'utilisateur de saisir une photo avec la caméra de l'appareil.
3.6.2.2. Le système doit convertir automatiquement l'image en format SVG.
3.6.2.3. Le système doit permettre à l'utilisateur de visualiser un aperçu des traits, tracés directement de façon superposée à la photo.
Le système doit permettre à l'utilisateur d'accéder à l'interface de construction d'un jeu à partir de l'accueil.
3.6.2.4. Le système doit permettre à l'utilisateur de choisir la langue (français ou anglais) du jeu.
3.6.2.5. Le système doit sélectionner la langue de l'utilisateur par défaut pour la création d'un jeu.
3.6.2.6. Le système doit permettre à l'utilisateur de saisir un mot à ajouter au jeu.
3.6.2.7. Le système doit assurer la validité du mot en le cherchant dans un dictionnaire anglais ou français.
3.6.2.8. Le système doit empêcher l'utilisateur d'ajouter un mot vulgaire en cherchant dans une liste de mots vulgaires.
3.6.2.9. Le système doit assurer l'unicité d'un mot.
3.6.2.10. Le système doit assurer que l'utilisateur soumet au moins un indice pour le jeu.
3.6.2.11. Le système doit permettre à l'utilisateur de soumettre le jeu si tous les champs sont correctement remplis.

3.7. Tutoriel [Essentielles]

3.7.1. Client lourd et léger

3.7.1.1. Le système doit permettre à l'utilisateur de suivre un tutoriel non interactif après la création de son compte, au premier démarrage de l'application.
3.7.1.2. Le système doit guider l'utilisateur à l'aide d'images représentant les différents outils et interfaces de l'application tout au long du tutoriel.
3.7.1.3. Le système doit permettre à l'utilisateur de progresser à travers les images du tutoriel tout au long du tutoriel.
3.7.1.4. Le système doit expliquer les différentes parties du tutoriel à l'utilisateur à l'aide de courtes phrases affichées à l'écran tout au long du tutoriel.
3.7.1.5. Le système doit permettre à l'utilisateur d'appuyer sur un bouton pour sauter le tutoriel.
3.7.1.6. Le système doit permettre à l'utilisateur de revoir le tutoriel en tout temps.

3.8. Tutoriel [Souhaitables]

3.8.1. Client lourd et léger

3.8.1.1. Le système doit permettre à l'utilisateur de suivre un tutoriel interactif après la création de son compte, au premier démarrage de l'application.
3.8.1.2. Le système doit permettre à l'utilisateur d'appuyer sur le bouton de l'interface mis en évidence pour progresser à travers le tutoriel.
3.8.1.3.
3.8.1.4. Le système doit expliquer chaque élément de l'interface à l'utilisateur à l'aide d'une infobulle pour chaque vue.

3.8.1.5. Le système doit guider l'utilisateur au travers de chaque vue essentielle.
Ces vues sont: l'accueil, la boîte de clavardage, la création d'un jeu, la création d'une partie, une partie mêlée générale, une partie sprint solo, une partie sprint coopératif, le profil.

3.9. Joueurs virtuels [Essentielles]

3.9.1. Le système doit s'assurer qu'un joueur virtuel puisse publier un message au début d'une partie.
Par exemple, le message peut être un message d'encouragement.
3.9.2. Le système doit s'assurer qu'un joueur virtuel puisse publier un message à la fin d'une partie.
3.9.3. Le système doit s'assurer qu'un joueur virtuel puisse publier un message à la fin de chaque tour.
3.9.4. Le système doit s'assurer qu'un joueur virtuel réponde à une demande d'indice d'un utilisateur.
3.9.5. Le système doit s'assurer qu'un joueur virtuel puisse dessiner lorsque vient son tour lors d'une partie mêlée générale.
3.9.6. Le système doit s'assurer qu'un joueur virtuel puisse dessiner lorsque vient son tour lors d'une partie sprint coopératif.
3.9.7. Le système doit s'assurer qu'un joueur virtuel puisse envoyer un message personnalisé selon les statistiques d'un utilisateur.
Les statistiques sont: historique des parties jouées, points de trophées, pourcentage de victoire.
3.9.8. Le système doit assigner une personnalité unique à un joueur virtuel qui se reflétera dans son style de dessin.
Par exemple, un joueur virtuel peut être colérique, comique ou triste.
3.9.9. Le système doit assigner une personnalité unique à un joueur virtuel qui se reflétera dans son message.

3.10. Système de succès [Souhaitables]

3.10.1. Client lourd et léger

3.10.2. Le système doit avertir un joueur par une notification lorsque celui-ci atteint un palier prédéfini sous différentes catégories de succès.
Ces catégories sont: le nombre de parties jouées, le nombre de parties gagnées, le nombre de mots devinés, le temps total passé sur l'application, le temps pour deviner un mot.
3.10.3. Le système doit assigner un nombre de "points de succès" prédéfini selon le palier et la catégorie du succès déverrouillé.
Par exemple, pour 1 partie gagnée, 5 points de succès sont accordés. Pour 5 parties gagnées, 10 points de succès sont accordés.
3.10.4. Le système doit permettre à chaque utilisateur de visualiser les points de trophée d'un autre utilisateur.
3.10.5. Le système doit permettre à l'utilisateur de visualiser ses succès déverrouillés à partir de son profil.

3.11. Clavardage [Essentielles]

3.11.1. Client lourd et léger

3.11.1.1. Le système doit permettre à l'utilisateur de clavarder dans une fenêtre intégrée à l'application.
3.11.1.2. Le système doit permettre à l'utilisateur de participer à plusieurs conversations en même temps.
3.11.1.3. Le système doit permettre à l'utilisateur de créer une conversation.
3.11.1.4. Le système doit permettre à l'utilisateur de rejoindre une conversation.

3.11.2. Client lourd

3.11.2.1. Le système doit permettre à l'utilisateur de clavarder dans un une fenêtre séparée de l'application.

3.11.3. Client léger

3.11.3.1. Le système doit notifier l'utilisateur à l'aide d'un indicateur visuel lors de la réception de message.

3.11.3.2. Le système doit notifier l'utilisateur à l'aide d'un effet sonore lors de la réception de message.

3.12. Clavardage [Souhaitables]

3.12.1. Client lourd et léger

3.12.1.1. Le système doit permettre d'afficher tout l'historique d'une conversation à un utilisateur.

3.13. Mode de jeu [Essentielles]

3.13.1. Client lourd et léger

3.13.1.1. Le système doit permettre à un dessinateur de dessiner un trait.
3.13.1.2. Le système doit permettre à un dessinateur d'effacer un trait au complet.
3.13.1.3. Le système doit permettre à un utilisateur d'effacer librement une partie du trait.
3.13.1.4. Le système doit permettre au dessinateur de sélectionner la pointe d'un trait.
3.13.1.5. Le système doit permettre au dessinateur de sélectionner la taille d'un trait.
3.13.1.6. Le système doit permettre au dessinateur de sélectionner la couleur d'un trait.
3.13.1.7. Le système doit ajouter un utilisateur à une conversation lorsque la partie débute.
3.13.1.8. Le système doit empêcher un joueur virtuel de deviner un mot.
3.13.1.9. Le système doit permettre à un joueur virtuel de tracer un dessin.
3.13.1.10. Le système doit modifier la vitesse de dessin d'un joueur virtuel en fonction de la difficulté d'un jeu.
Par exemple: le joueur virtuel prend 5 secondes à dessiner un dessin en mode facile, 10 secondes en mode moyen et 20 secondes en mode difficile.
3.13.1.11. Le système doit permettre à un seul joueur de dessiner en mode mêlée générale.
*Les exigences suivantes concernent le mode **mêlée générale**:*
3.13.1.11.1. Le système doit permettre à un joueur, qui regarde un dessin, d'écrire le mot correspondant au dessin.
3.13.1.11.2. Le système doit donner des points à un joueur qui a deviné le mot correspondant au dessin affiché.
3.13.1.11.3. Le système doit jouer un effet sonore lorsque le mot est deviné par un joueur.
3.13.1.11.4. Le système doit donner des points à un joueur qui dessine en fonction du nombre de joueurs ayant deviné le mot correspondant au dessin.
3.13.1.11.5. Le système doit changer de dessinateur à chaque 60 secondes.
3.13.1.11.6. Le système doit changer de dessinateur si tous les autres joueurs ont deviné le mot correspondant au dessin.

3.14. Mode de jeu [Souhaitables]

3.14.1. Client lourd et léger

3.14.1.1. Le système doit permettre à un utilisateur de participer à une partie en mode sprint solo.
*Les exigences suivantes concernent le mode **sprint solo**:*
3.14.1.1.1. Le système doit commencer la partie avec un temps de 60 secondes.
3.14.1.1.2. Le système doit terminer la partie une fois le temps écoulé.
3.14.1.1.3. Le système doit changer le dessin une fois le nombre d'essais maximum pour deviner les mots est atteint.
3.14.1.1.4. Le système doit définir le nombre d'essais maximum selon la difficulté d'un jeu.
Par exemple: Essais illimités pour un jeu facile, 10 essais pour un jeu moyen et 3 essais pour un jeu difficile.
3.14.1.1.5. Le système doit diminuer de 1 la quantité d'essais restante pour chaque tentative de mot échouée.
3.14.1.1.6. Le système doit afficher le temps restant à la partie en mode solo.
3.14.1.1.7. Le système doit afficher le nombre d'essais restant au joueur pour le dessin en cours.
3.14.1.1.8. Le système doit afficher le nombre de dessins deviné par le joueur pour la partie en

cours.

3.14.1.1.9. Le système doit ajouter du temps à la partie en fonction du niveau de difficulté du dessin lorsque le mot est trouvé.
Par exemple: 10 secondes ajoutées pour un jeu facile, 5 secondes pour un jeu moyen et 0 seconde pour un jeu difficile.

3.14.1.1.10. Le système doit ajouter un point au joueur lorsqu'un mot est trouvé.

3.14.1.2. Le système doit permettre à un utilisateur de participer à une partie en mode sprint coopératif.
*Les exigences suivantes concernent le mode **sprint coopératif**:*

3.14.1.2.1. Le système doit commencer la partie avec un temps de 60 secondes.

3.14.1.2.2. Le système doit terminer la partie une fois le temps écoulé.

3.14.1.2.3. Le système doit définir le nombre d'essais maximum collectif selon la difficulté d'un jeu.
Par exemple: Essais illimités pour un jeu facile, 10 essais pour un jeu moyen et 3 essais pour un jeu difficile.

3.14.1.2.4. Le système doit changer le dessin une fois le nombre d'essais maximum pour deviner les mots est atteint.

3.14.1.2.5. Le système doit afficher le temps restant à la partie.

3.14.1.2.6. Le système doit afficher le nombre d'essais restant à l'équipe pour le dessin en cours.

3.14.1.2.7. Le système doit afficher le nombre de dessins deviné par l'équipe pour la partie en cours.

3.14.1.2.8. Le système doit ajouter du temps à la partie en fonction du niveau de difficulté du dessin lorsque le mot est trouvé.
Par exemple: 10 secondes ajoutées pour un jeu facile, 5 secondes pour un jeu moyen et 0 seconde pour un jeu difficile.

3.14.1.2.9. Le système doit ajouter un point à l'équipe lorsqu'un mot est trouvé.

3.14.1.2.10. Le système doit diminuer de 1 la quantité d'essais restante à l'équipe pour chaque tentative de mot échouée.

3.15. Traduction [Essentielles]

3.15.1. Client lourd et léger

3.15.1.1. Le système doit pouvoir traduire l'interface utilisateur de l'application en fonction de la langue du système (Anglais ou Français) lors de la première utilisation.

3.15.1.2. Le système doit pouvoir traduire l'interface utilisateur de l'application grâce à un bouton (Anglais ou Français).

3.15.1.3. Le système doit séparer un joueur francophone d'un joueur anglophone dans un lobby.

3.15.1.4. Le système doit traduire le message envoyé par un joueur virtuel durant une partie.

3.16. Vue administrateur [Essentielles]

3.16.1. Client lourd

3.16.1.1. Le système doit permettre à un administrateur d'avoir accès à la vue administration.

3.16.1.2. Le système doit avoir utilisateur Super Admin.

3.16.1.3. Le système doit permettre à un administrateur de donner les permissions d'administration à un utilisateur non-admin.

3.16.1.4. Le système doit permettre à un administrateur de supprimer un compte utilisateur.

3.16.1.5. Le système doit permettre à un administrateur de changer le mot de passe d'un utilisateur non-admin.

3.16.1.6. Le système doit afficher le temps de fonctionnement du serveur.

3.16.1.7. Le système doit afficher le pourcentage d'utilisation du processeur du serveur.

3.16.1.8. Le système doit afficher la quantité de mémoire utilisée par le serveur.

3.16.1.9. Le système doit afficher le nombre de clients sur l'application.

3.16.1.10. Le système doit afficher le nombre total de comptes utilisateur.

3.17. Vue administrateur [Souhaitables]

3.17.1. Client lourd

3.17.1.1. Le système doit permettre de voir tous les jeux disponibles sur l'application
3.17.1.2. Le système doit permettre d'effacer un jeu de l'application
3.17.1.3. Le système doit permettre de modifier un indice d'un jeu
3.17.1.4. Le système doit permettre de modifier le mot d'un jeu

4. Exigences non fonctionnelles

4.1. Utilisabilité

4.1.1. Facilité d'utilisation

Assure que le logiciel est simple d'utilisation

4.1.1.1 Le temps de formation requis à un utilisateur normal est de 5 minutes au maximum.
4.1.1.2 Le temps de formation requis à un utilisateur spécialisé est de 3 minutes au maximum.
4.1.1.3 Les messages d'erreurs doivent proposer des solutions en cas d'erreurs.

4.1.2 Temps des tâches

Les exigences présentées ici sont nécessaires afin que le logiciel possède une bonne utilisabilité.

4.1.2.1 La création d'une nouvelle partie doit se faire en moins de 1 minute.
4.1.2.4 La création d'un nouveau compte doit prendre moins de 1 minute et 30 secondes.
4.1.2.5 La connexion au compte doit se faire en moins de 15 secondes.
4.1.2.6 L'utilisateur doit pouvoir utiliser le clavardage en moins de 1 minute.
4.1.2.7 L'utilisateur doit pouvoir consulter son profil en moins de 10 secondes.
4.1.2.8 La durée du tutoriel doit être de moins de 2 minutes.
4.1.2.9 L'utilisateur doit changer le thème de l'application en moins de 15 secondes.

4.1.3 Interface utilisateur

L'interface utilisateur doit respecter les exigences suivantes.

4.1.3.1 Les éléments de l'interface utilisateur de chaque plateforme doivent utiliser le même style et thème.
4.1.3.2 La différence du temps d'une complétion d'une tâche effectuée sur le client léger et le client lourd doit être au maximum de 10%.
4.1.3.3 L'interface du canevas de dessin doit occuper 60% de l'écran.

4.2. Fiabilité

4.2.1. Disponibilité du système

4.2.1.1 Le système doit être disponible 99.9% du temps par année.
4.2.1.2 Le temps moyen entre pannes du système est de six mois.
4.2.1.3 Le temps moyen d'une panne du système est de trois heures.

4.3. Performance

4.3.1. Performance du service

4.3.1.1 Le temps requis pour recevoir le message d'un autre joueur sur un canal de discussion est d'une seconde au maximum.
4.3.1.2 Le temps requis pour recevoir un message du serveur sur un canal de discussion est d'une seconde au maximum.
4.3.1.3 Le temps requis pour que le système valide le nom d'utilisateur et le mot de passe d'un utilisateur est d'une seconde au maximum.
4.3.1.4 Le serveur doit fournir une expérience de jeu sans latence.
4.3.1.5 Le serveur doit supporter simultanément la connexion de 8 utilisateurs.
4.3.1.6 Un utilisateur peut accéder au profil d'un utilisateur en 1.5 seconde au maximum.
4.3.1.7 Le temps requis pour créer un compte doit être moins de 1.5 seconde.
4.3.1.8 Le temps requis pour créer un jeu doit être de moins de 1.5 seconde .
4.3.1.9 Le temps requis pour tracer une image en vectoriel doit être moins de 1.5 seconde.

4.3.2 Performance du client léger

4.3.2.1 Le client léger doit pouvoir s'ouvrir en moins de 2 secondes.
4.3.2.2 Les traits dans une partie doivent pouvoir se dessiner aux alentours de 30fps.
4.3.2.3 Le client léger doit pouvoir changer le thème de l'interface sans ajouter de délai aux autres tâches.

4.3.3 Performance du client lourd

4.3.3.1 Le client lourd doit pouvoir s'ouvrir en moins de 3 secondes.
4.3.3.2 Les traits dans une partie doivent pouvoir se dessiner aux alentours de 30fps.

4.4. Maintenabilité

4.4.1. Normes de codage

4.4.1.1 Le code doit être écrit en anglais.
4.4.1.2 Chaque répertoire à la racine du projet doit être nommé en PascalCase.
4.4.1.3 Chaque fichier à la racine du projet doit être nommé en PascalCase.
4.4.1.4 Le client lourd (C#) doit suivre les conventions de Microsoft sur C# disponibles à https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/coding-conventions
4.4.1.5 Le client léger (Kotlin) doit suivre les conventions de la Kotlin Foundation disponibles à https://kotlinlang.org/docs/reference/coding-conventions.html
4.4.1.6 Le serveur (Go lang) doit utiliser les conventions de Google sur le Go disponibles à https://golang.org/doc/effective_go.html
4.4.1.7 Chaque projet (client lourd, léger et serveur) doit être équipé d'un "linter".

4.4.2 Réutilisation de code

4.4.2.1 Le GUI doit être modulaire afin de faciliter sa réutilisation.
4.4.2.2 Le couplage doit être le plus faible possible entre les composants du logiciel dans le souci de réutilisation de code.
4.4.2.3 La documentation doit accompagner ces composants afin de s'assurer qu'ils sont facilement réutilisables.

4.4.3 Intégration continue

4.4.3.1 Chaque artefact exécutable doit avoir un numéro de version associée qui correspond aux différents jalons.
4.4.3.2 Chaque artefact exécutable doit utiliser les étiquettes "tags" pour indiquer les versions dans le contrôle de version.
4.4.3.3 Chaque artefact exécutable doit avoir de l'intégration continue qui s'occupe de faire la gestion des

artefacts

4.4.3.4 L'intégration continue doit compiler le code.

4.4.3.5 L'intégration continue doit linter le code pour vérifier qu'il n'a pas d'erreurs aux conventions.

4.4.3.5 Le serveur doit être déployé lorsque des changements sont faits sur certaines branches du système de contrôle de version.

4.5. Contraintes de conception

4.5.1. Langage et plateforme

Langage et plateforme utilisés pour le développement du logiciel.

4.5.1.1 Le langage de programmation du client léger doit être le Kotlin.

4.5.1.2 Le langage de programmation du client lourd doit être le C#.

4.5.1.3 Le langage de programmation du serveur doit être le Golang.

4.5.1.4 L'environnement de développement du client léger doit être Android Studio 3.5.2.

4.5.1.5 L'environnement de développement du client lourd doit être Rider 2019.3.1.

4.5.1.6 L'environnement de développement du serveur doit être VS Code 1.41

4.5.1.7 La version du cadriciel .NET utilisée par le client lourd doit être 4.7.2.

4.5.2 Gestion des libraires de classes

Exigences pour la gestion des libraires de classes.

4.5.2.1 Chaque libraire de classes doit être approuvée par l'équipe avant d'être introduite dans le projet.

4.5.2.2 Le logiciel doit minimiser le nombre de dépendances aux libraires de classes moins de 25.

4.5.2.3 Le client lourd doit utiliser NuGet pour la gestion des dépendances.

4.5.2.4 Le client léger doit utiliser Gradle.

4.5.2.5 Le client léger doit gérer les dépendances avec Maven.

4.5.2.6 Le serveur doit utiliser Go pour gérer les dépendances. Commande *go get*

4.6. Sécurité

4.6.1 Confidentialité de l'information

4.6.1.1 Chaque compte collecte des informations de l'utilisateur.

Les informations sont les suivantes: nom d'utilisateur, courriel, mot de passe (hashé), nom et prénom.

4.6.1.2 L'information recueillie doit être pour une utilisation interne seulement et pour offrir le service.

4.6.1.3 Aucune information ne doit être utilisée pour des annonceurs ou pour des fins commerciales.

4.6.1.4 Les journaux doivent être libres d'informations confidentielles.

4.6.1.5 Les journaux ne doivent pas contenir de secret comme des mots de passe.

4.6.2 Mots de passe

4.6.2.1 Les mots de passe doivent respecter les critères de la norme du NIST soit NST 800-63 rev3

4.6.2.2 Les mots de passe doivent être validés contre une liste de mots de passe communs.

4.6.2.3 Les mots de passe doivent être cryptés à l'aide de l'algorithme bcrypt dans la base de données pour assurer leur sécurité et leur confidentialité.

4.6.2.4 Les échanges de mots de passe sur le réseau doivent être faits qu'une seule fois pour récupérer un jeton.

4.6.2.5 Les jetons sont ensuite utilisés pour authentifier les utilisateurs déjà connectés.

4.6.2.6 Les jetons sont utilisés pour toutes les autres requêtes qui nécessitent de l'authentification.

4.6.2.7 Le système doit bloquer l'utilisateur pour 5 minutes après 10 tentatives.

4.6.3 Gestion des jetons

4.6.3.1 Les jetons doivent être générés avec des fonctions de génération d'octets aléatoire avec une entropie élevée.

4.6.3.2 Les jetons doivent être enregistrés dans une voûte de gestion de jetons.

4.6.3.3 Le client lourd doit utiliser le Windows Credential API pour sauvegarder le jeton.

4.6.3.4 Le client léger doit utiliser Credential API d'Android pour sauvegarder le jeton.
4.6.3.5 Le jeton doit être unique pour chaque client.
4.6.3.6 Les jetons doivent être sauvegardés dans une base de données sur le serveur.
4.6.3.7 Les jetons et mots de passe doivent être masqués lorsque les données de productions doivent entrer dans un environnement de développement.

4.7. Exigences de la documentation usager en ligne et du système d'assistance

4.7.1 Système d'assistance

4.7.2.1 Le système doit avoir un tutoriel pour expliquer les fonctionnalités de l'application.

4.8. Normes applicables

4.8.1 La norme NIST 800-63 rev 3. doit être suivie pour la sécurité des mots de passe.