# Documentation project ludum :

## Comptes utilisateurs :

Les comptes sont stocké dans un fichier csv servant de base de données. La méthode « CreateAccount » permet d’ajouter un compte dans la base de donnée d’après un pseudo, une adresse mail et un mot de passe. On écrit dedans en utilisant console writeLine à partir d’un certain format qui contient l’ancienne sauvegarde de la base de donnée pour éviter de tout effacer chaque fois qu’on écrit dessus.   
Au moment de se créer un compte son mot de passe est hashé grâce à la fonction « HashPassword » trouvé sur internet utilisant BouncyCastle, le code trouvé au lien : <https://stackoverflow.com/questions/57857330/how-can-i-get-sha3-512-hash-in-c> par l’utilisateur NineBerry.  
Finalement pour se connecter on utilise la méthode ConnectAccount qui retourne true si on arrive à se connecter ou false si la connexion échoue. Pour vérifier ceci on lie chaque ligne une à une, puis si la ligne qu’on li correspond au mot de passe et au mail on indique que l’utilisateur est connecté, sinon on attend de lire tout le document pour retourner que l’utilisateur n’a pas pu se connecter.

## Gestion des amis et des groupes :

Les amis sont stockés dans un objet ami qui garde leur nom ainsi que leur date d’ajout, Les amis sont ensuite stocké dans un objet liste d’ami qui permet d’interagir sur eux, les supprimer, les ajouter et accéder à leurs informations. Les groupes sont eux aussi des objets qui sont stocké dans une liste de groupe qui permet d’ajouter des groupes d’après des amis, de supprimer des groupes et de voir les groupes.

## Distributions des cartes :

Pour distribuer les cartes j’ai créé en premier lieu un objet carte qui ne contient rien mise à part un index pour vérifier que les cartes soient mélangées et un objet joueur qui aura une liste de carte comme main. Dans le programme on initialise en premier lieu un deck de carte qui n’est qu’une liste puis on initialise les cartes avant de lancer la méthode « blend ». La méthode retournera les cartes mélangées dans un ordre aléatoire en créant une seconde liste qui stockera les cartes une fois mélangé. On retire à chaque fois une carte aléatoirement dans le paquet de carte initiale pour les mettre dans la seconde liste puis renvoyé la liste.

## I18N:

J'ai rempli deux fichiers de ressources avec les traductions en anglais et en Français des mots qui seront utilisés dans le jeu.

Les strings pour accéder à ces mots/phrases via le ressource manager sont presque les mêmes mots en anglais mais simplifiés (pas d'espace ni d'accent, en CamelCase).

Ce qui m'a pris pas mal de temps a été de trouver un moyen d'optimiser le changement des langues, d'abord si je cliquais sur le bouton Français tous les autres boutons changeait de langues, de même pour le bouton anglais.

Pour optimiser tout cela j'ai décidé de créer une methode "ChangeLanguage". Dans celle-ci j'ai d'abord pensé vouloir changer la valeur du Typeof dans "rm = new ResourceManager(typeof(FR))" soit FR soit EN avec des ifs...

Cela n'a pas marché étant donné que les typeof n'ont pas l'air d'aimer d'être changés avec des variables...

Finalement après des recherches et quelques tests j'ai trouvé que le switch était en fait le plus approprié. Ainsi la version finale du changement de la langue se fera avec un switch.

Les boutons Anglais et Français sont des boutons avec une simple image étirée du drapeau de la langue. J’ai trouvé le style "FlatStyle" plus approprié pour ces boutons étant donné que si la souris passait en dessus d'eux ils ne montraient pas forcément qu'ils etaient clickables... Cela ajoute une sorte de 3D bien agréable.

## Serveur :

Ajout du système de lecture du fichier de configuration pour lancer le jeu ainsi que système pour savoir si le serveur principal est actif. Réalisation d'une application CSharp qui permet d'envoyer et de recevoir des messages depuis le broadcast. Implémentation de l'ouverture de fichier en Rust. Recherche sur le web pour savoir comment ouvrir des fichiers avec Rust  
Lien : <https://doc.rust-lang.org/std/process/struct.Command.html>  
Annexe : [Image](Uno-server-system.png)

# Rapport test :

## Comptes utilisateurs :

J’ai créé un utilisateur dans la base de donnée puis j’ai vérifié que je pouvais me connecter à cet utilisateur, et finalement j’ai vérifié dans la base de donnée que le mot de passe pour créer cet utilisateur était hashé.

## Gestion des amis et des groupes :

J’ai créé une application windows form pour vérifier les critéres d’acceptations. Grace à cette application j’ai pu vérifier qu’un groupe ne pouvait pas posséder plus de 7 amis, moins de 2, que les groupes peuvent être supprimé et que des joueurs peuvent être retiré et ajouté au groupe. Grace à pseudo animation j’ai aussi pu ajouter et retiré des amis de la liste d’amis et j’ai vérifié à l’aide des points d’arrêt que les amis aient un pseudo et une date d’ajout.

## Distributions des cartes :

J’ai vérifié en mettant des points d’arrêts que chaque objet joueur possédait 6 cartes et que chaque carte avait été enlevé du deck quand elles étaient distribuées. J’ai vérifié que les cartes étaient mélangées aléatoirement dans le deck en vérifiant l’index de l’objet carte.