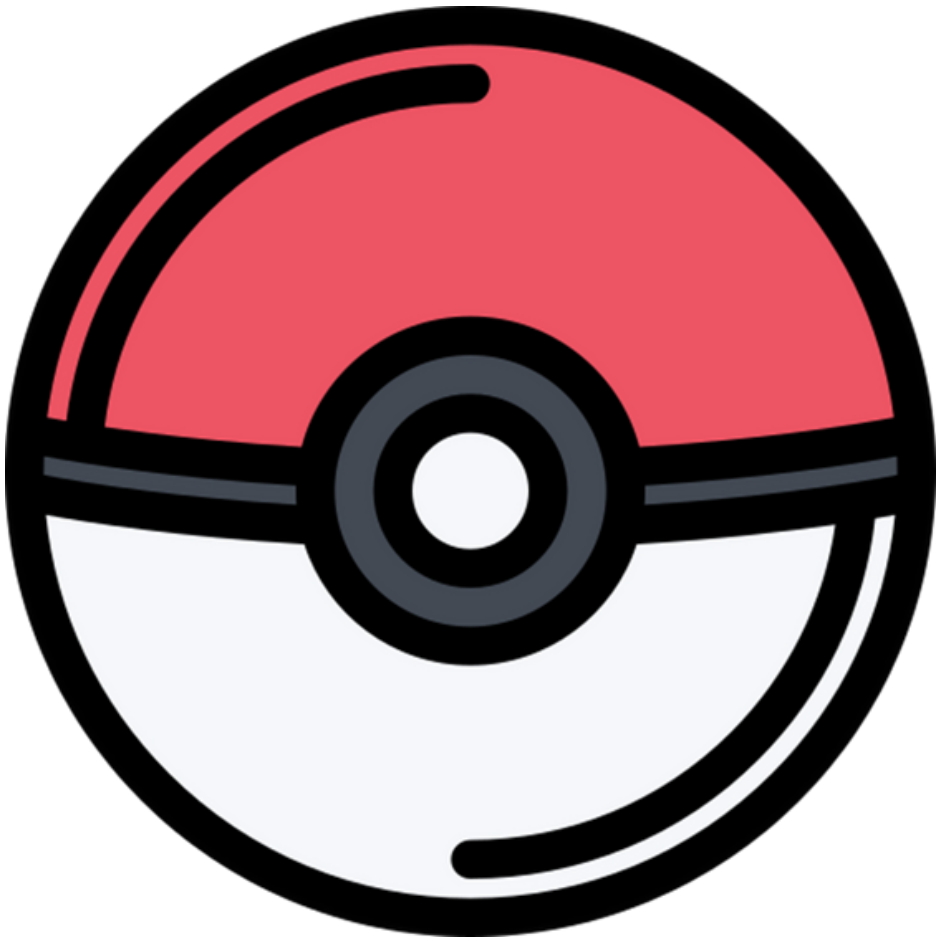
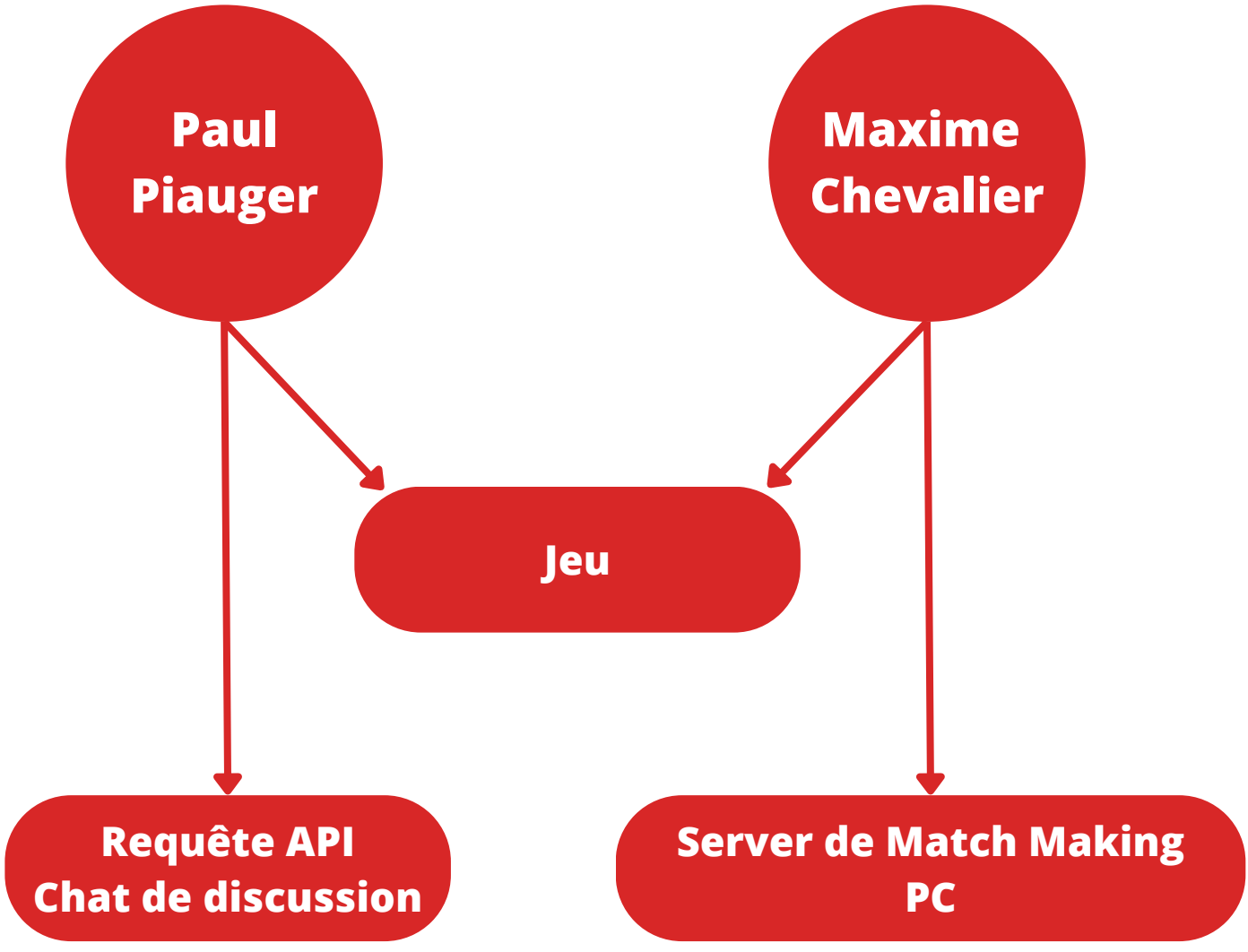
Projet Développement
Poké-Hide



Sommaire

Rôle


Paul
Piauger
Maxime
Chevalier
Jeu
Requête API
Chat de discussion
Server de Match Making
PC

Technologie Utilisé

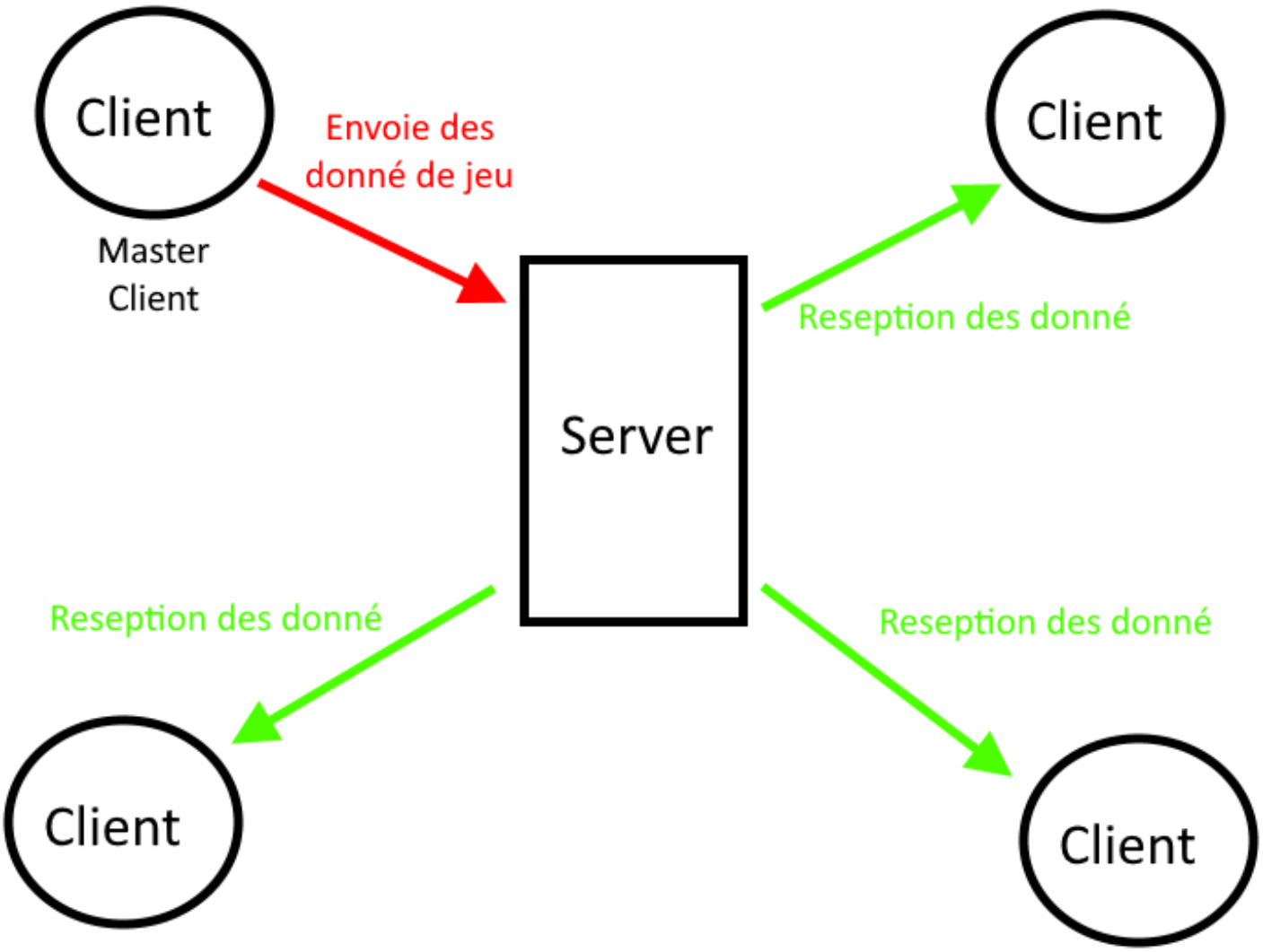
Unity :
Moteur de Jeu et interface de développement

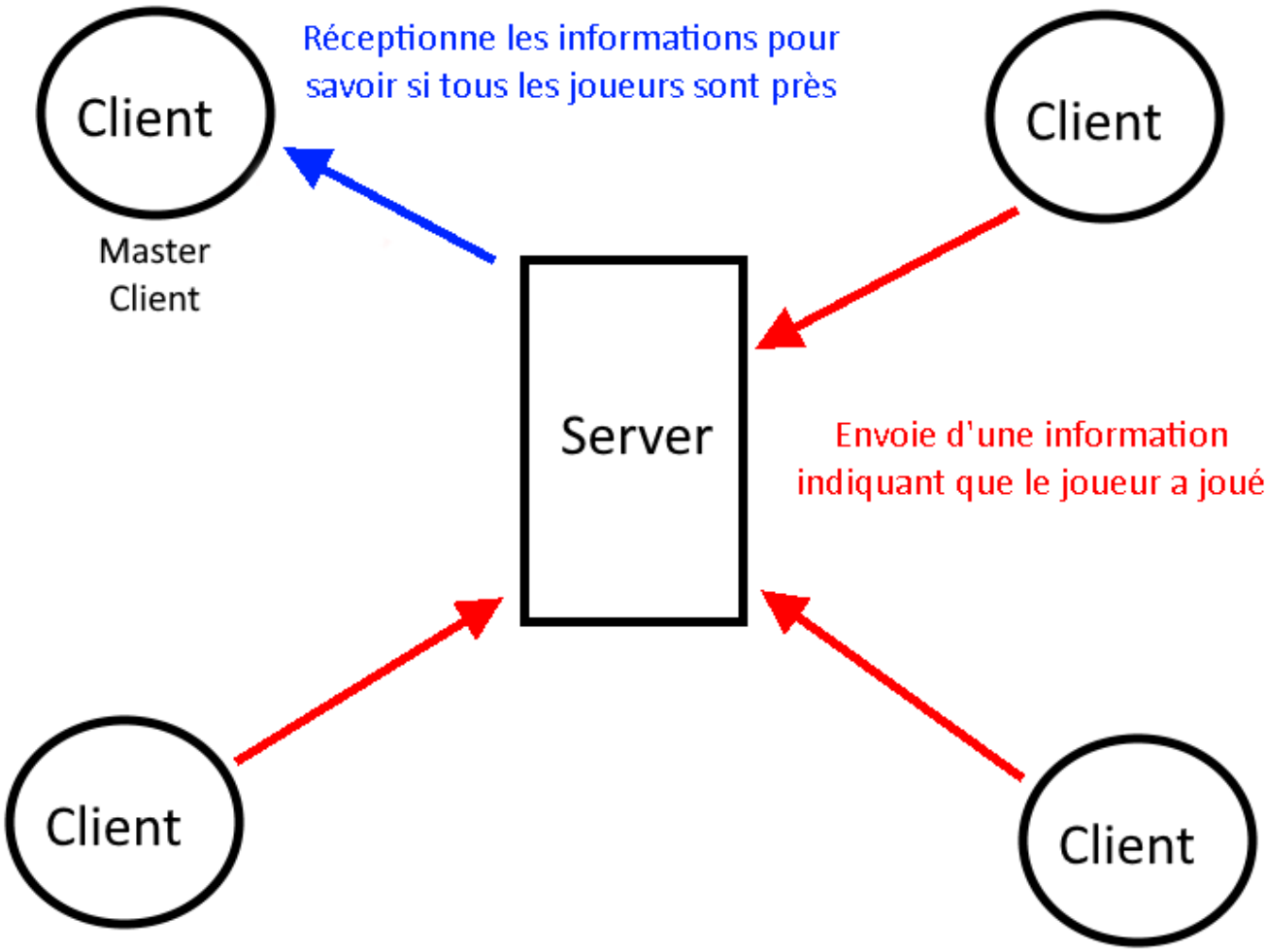
C# :
Language de programmation orienté objet

SQLite :
Base de donné basé sur SQL

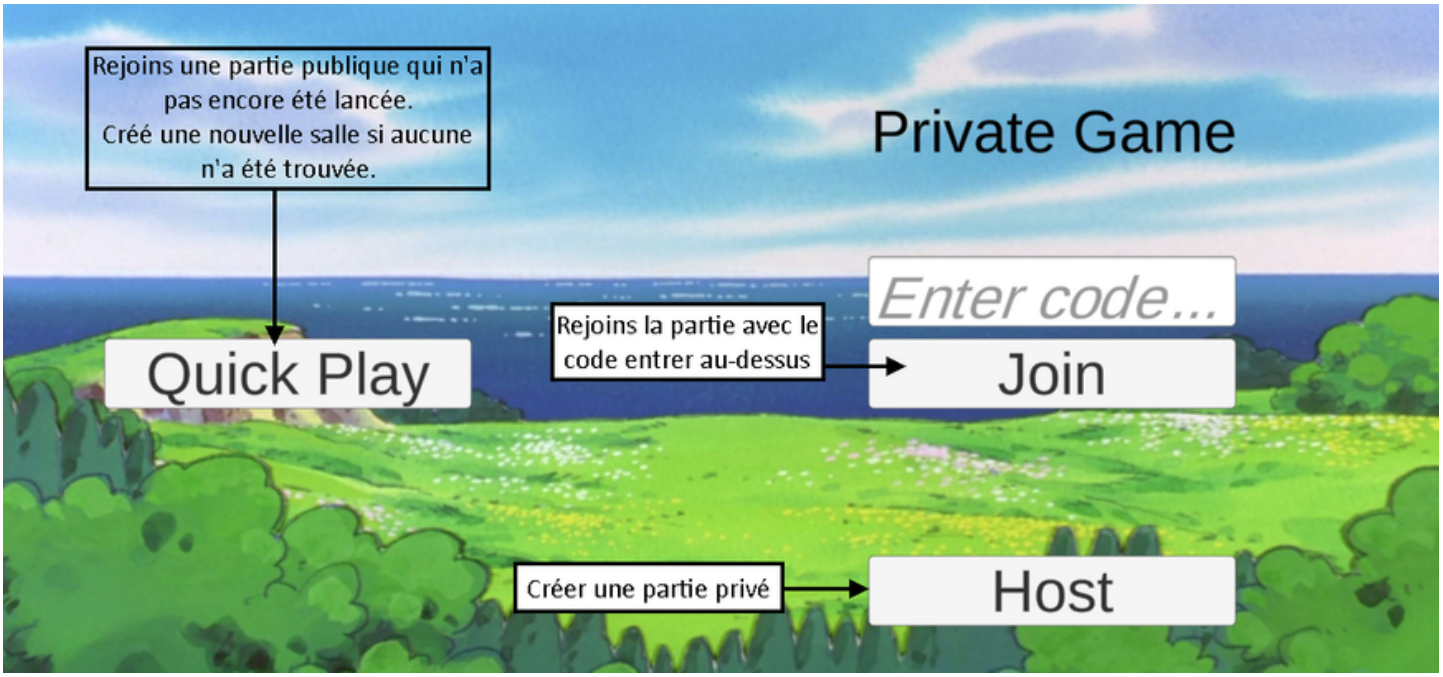
Photon :
Ebergement de server de Jeu pour Unity

Relation Server et client




Client
Master
Client
Réceptionne les informations pour
savoir si tous les joueurs sont près
Client
Server
Envoie d'une information
indiquant que le joueur a joué
Client
Client

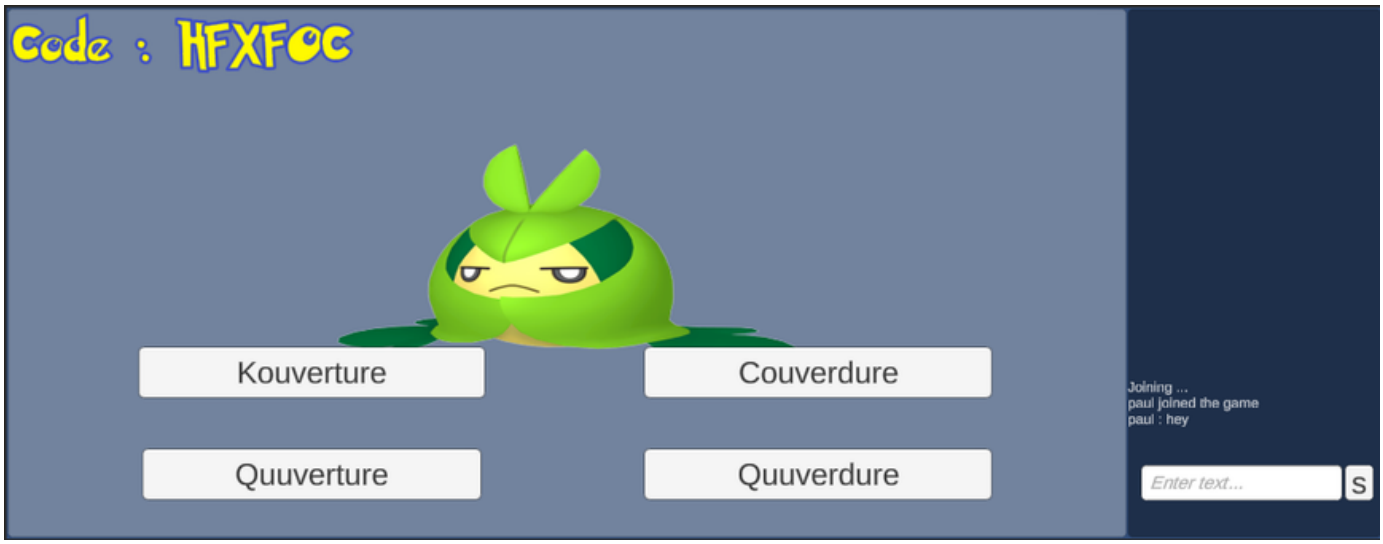
Gestion des salons



Au moment où le joueur souhaite rejoindre un salon de jeu, 3 options s'offrent à lui :

- Quick Play : si le joueur choisit cette option, le serveur va chercher si des parties publiques sont encore ouvertes et va faire rejoindre le joueur à celle-ci. Si aucun salon n'est trouvé, il en créera une automatiquement.
- Join : le joueur pourra entrer un code dans la case situé juste au-dessus afin de rejoindre le salon correspondant à ce code. Cela permet de pouvoir rejoindre des amis. Le code se situe en haut à gauche d'un salon.
- Host : le joueur pourra créer un salon privé ou le seul moyen d'y entrer est de passer par le code de la salle.

Fonctionnement du jeu



Une fois le jeu commencé, une image de Pokémon apparaît sur l'écran et 4 noms possibles pour celui-ci apparaissent. Une fois le jeu commencé, une image de Pokémon apparaît sur l'écran et 4 noms possibles pour celui-ci apparaissent. Il suffit de cliquer sur un nom pour sélectionner et validé la réponse. Dans le cas où une mauvaise réponse serait donnée, 250 points sont retirés au joueur. Sinon, si une bonne réponse est sélectionnée un certain nombre de points est distribué en fonction du temps de réponse (1000 points si répondus en 0 sec, 500 si répondus en 5 sec, 50 si répondus en 9,5 sec). Dans le cas où aucune réponse ne serait entrée au bout de 10 sec, le jeu considère cela comme une erreur. Une fois répondu un message est envoyer au serveur pour indiquer qui a joué. Ensuite, nous attendons que tous les joueurs aient joué pour passer à la manche suivante.

Récupération des Pokémon

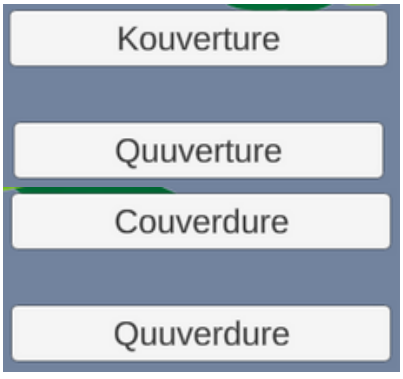
Pour générer les image présente dans le jeu, nous faisons une raquette à l'api. Il faut savoir que chaque Pokémon a un id propre et chaque id est différent. Nous avons donc fait une fonction qui génère un nombre aléatoire compris dans la liste des id Pokémon disponible dans l'api. Nous renseignons ensuite le chiffre dans la requête pour récupérer toutes les informations pour le Pokémon correspondant à l'id. Grace a une autre fonction, on vient séparer toutes les informations pour les rendre disponibles via une variable. Puis avec la variable contenant le lien de l'image nous l'envoyé à Unity pour qu'il change l'image de base en la remplaçant par celle de notre lien.

```
public IEnumerator GetImage(string imageUrl)
{
    GameManager.instance.SetReady(false);
    UnityWebRequest request = UnityWebRequestTexture.GetTexture(imageUrl);
    yield return request.SendWebRequest();
    if (request.isNetworkError || request.isHttpError)
    {
        Debug.Log(request.error);
    }
    else
    {
        Texture2D texture = DownloadHandlerTexture.GetContent(request) as Texture2D;
        GetComponent<Image>().sprite = Sprite.Create(texture, new Rect(0, 0, texture.width, texture.height), new Vector2(0, 0));
        GameObject panel = GameObject.Find("GamePanel");
        GameManager.instance.SetReady(true);
        GameManager.instance.timeToWait = 0;
        while (!GameManager.instance.AllPlayerReady()) yield return new WaitForSeconds(1);
        for (int i = 0; i < 4; i++)
        {
            panel.transform.GetChild(i + 1).GetComponent<Button>().enabled = true;
            panel.transform.GetChild(i + 1).transform.GetChild(0).GetComponent<TextMeshProUGUI>().enabled = true;
        }
        GameManager.instance.time = 0f;
        GameManager.instance.isTime = true;
        GameManager.instance.SetReady(false);
        GameManager.instance.cache.SetActive(false);
    }
}
```

Génération des noms

Pour générer les différents noms de Pokémon a utilisé comme réponse possible. Nous récupérons dans un premier temps le nom officiel du Pokémon en question que nous utilisons comme base pour la création des variantes. Ensuite, nous modifions aléatoirement certaine lettre selon la table ci-dessous.
Une fois générer, le programme vérifie si ce nouveau nom n'est pas déjà utilisé. Dans ce cas, il est sauvegardé et près a être utiliser, sinon un nouveau nom est générer jusqu'à un maximum de 10 fois qui au bout du 10e s'il est différent du nom officiel est gardé bien qu'il soit déjà utilisé. Cela a pour but de ne pas bloquer le programme dans le cas où le nom ne pourrait pas avoir plus de 3 variantes.

table ds chagement de lettre
o <=> a
k <=> c <=> qu
d <=> t



Chat en jeu

Dans notre jeu, un système de chat est disponible, le jeu va l'afficher au plus bas de la zone de texte du chat et faire remonter tous les anciens messages. En bas de celui-ci, on peut voir une zone dans laquelle l'utilisateur peut entrer son message, puis l'envoyé via le bouton situé sur sa droite. Celui-ci va ensuite être brodcaster a tous les joueurs connecter au salon en ajoutant à celui-ci un préfixe composé du pseudo du joueur suivie de ":". À la réception d'un message en provenance du serveur. Le jeu va l'afficher au plus bas de la zone de texte du chat et faire remonter tous les anciens messages. En bas de celui-ci, on peut voir une zone dans laquelle l'utilisateur peut entrer son message, puis l'envoyé via le bouton situé sur sa droite. De plus, le serveur enverra automatiquement un message indiquant qu'un joueur vient de rejoindre le salon.

Gestion du pc

À la fin d'une partie, la moité supérieur se voit capturer un Pokémon qui sera ajouté à son PC, le PC, dans l'univers Pokémon, c'est l'emplacement ou un dresseur peut stocker tous ses Pokémon. Ici, notre PC est une table SQL stockée en local par chacun des joueurs. Un id aléatoire est généré pour décider quel Pokémon a été capturer. Le joueur pour y avoir accès et collectionner sous forme de carte la liste des Pokémon.

```
public void addPokemon( int pokedexId, int generation, string name, string category, string sprite, string type1, string type2){

        using (var conn = new SqliteConnection(DB))
        {
            conn.Open();
            using (var cmd = conn.CreateCommand())
            {
                cmd.CommandText = "INSERT INTO PC (pokedexId, generation, name, category, sprite, type1, type2) VALUES ('" + pokedexId + "', '" +
generation + "', '" + name + "', '" + category + "', '" + sprite + "', '" + type1 + "', '" + type2 + "')";
                cmd.ExecuteNonQuery();
            }
            conn.Close();
        }

    }
```