

# FICHE : PoC & CHOIX TECHNOLOGIQUES

=> Après avoir rempli cette fiche :convertir ce document en PDF et le téléverser dans votre dépôt Github dans un répertoire nommé 'doc'.

## COMPTE RENDU DE LA VEILLE TECHNOLOGIQUE PoC

### QUESTIONS DE RECHERCHE SUR LA PoC

Quel est l'opération la plus risquée de votre projet, l'élément qui risque de ne pas fonctionner ? Quel est le PROBLÈME à résoudre ?

1-La communication entre deux clients Unity par un serveur Node.JS.

2-Multijoueur en 1vs1

Quelle est la QUESTION que vous vous posez et que vous demandez à internet de répondre ?

1- Comment faire communiquer deux clients Unity à travers un serveur Node.JS ?

2- Comment faire pour limiter une partie multijoueur à 2 joueurs ?

### POC = PREUVE DE CONCEPT

Quel genre de preuve de concept pourrait valider que le problème n'existe pas ou qu'une solution a été trouvée ?

1-Faire connecter 2 clients Unity ensemble sur le serveur pour envoyer un message JSON à l'autre client Unity.

2-Faire connecter 3 clients Unity ensemble sur le serveur et en mettre 2 en joueur et 1 en spectateur, 1 des joueurs doit envoyer un message JSON et les 3 clients Unity doivent recevoir le message JSON.

### LES MARQUE-PAGES IDENTIFIÉS LORS DE VOS RECHERCHES

Lien vers une page publique contenant vos marque-pages collaboratifs ou lister les marque-pages directement ici. Pour chaque lien : URL, nom de la page et description sommaire.

Documentation de Colyseus.io : <https://docs.colyseus.io/colyseus/> , framework pour faire des serveurs en Node.JS

Playlist d'un tutorial pour un jeu multijoueur utilisant Colyseus.io :

<https://www.youtube.com/playlist?list=PLumYWZ2t7CRueXsocQXOGgewmwzohljof>

# LES PREUVES DE CONCEPT

Pour chaque preuve de concept réalisée : identifier le but de la preuve de concept (ce qu'elle vérifie), le lien vers le sous-répertoire de votre dépôt GitHub qui contient le code de la preuve de concept ainsi que les résultats de votre expérimentation, puis, finalement, vos conclusions.

## PREMIÈRE POC RÉALISÉE

**Preuve :** Faire communiquer 2 clients Unity par un serveur Node.JS

**URL Github :**

## EXPLIQUEZ VOTRE POC

### Décrivez la Poc en détails.

Faire en sorte que 2 clients Unity peuvent se connecter sur le serveur Node.JS

Vérifier que les 2 clients Unity sont connectés sur le même serveur.

Vérifier que les 2 clients Unity peuvent envoyer des messages au serveur

Vérifier que les 2 clients Unity peuvent recevoir les messages depuis le serveur

### Que PROUVE la Poc ?

La PoC prouve que plusieurs clients Unity peuvent communiquer sur le même serveur

### Que reste-t-il à prouver ?

Sachant que je veux faire que du multijoueur en 1vs1, il faudrait prouver qu'il n'y a que 2 joueurs réellement et si il y a plus de personnes se connectant il faudrait qu'ils deviennent spectateurs.

### Quels sont vos résultats de la PoC ?

Les 2 clients peuvent envoyer des messages "Hello" en JSON au serveur

Le serveur redirige les messages à tous les clients en JSON

## DEUXIÈME POC RÉALISÉE

**Preuve :** Connecter 3 clients Unity ensemble sur le serveur et en mettre 2 en joueur et 1 en spectateur

**URL Github :**

## EXPLIQUEZ VOTRE POC

### Décrivez la Poc en détails.

Faire en sorte que 3 clients Unity peuvent se connecter sur le serveur Node.JS

Vérifier que les 3 clients Unity sont connectés sur le même serveur.

Vérifier que les 2 clients Unity peuvent envoyer des messages au serveur

Vérifier que les 3 clients Unity peuvent recevoir les messages depuis le serveur

### Que PROUVE la Poc ?

La PoC prouve que seulement 2 clients Unity peuvent interagir avec le serveur et que le 3eme client Unity ne peut que recevoir des informations du serveur.

### Que reste-t-il à prouver ?

Rien

### Quels sont vos résultats de la PoC ?

Les 2 clients peuvent envoyer des messages "Hello" en JSON au serveur

Le 3e clients ne peut pas envoyer de message quel qu'il soit au serveur

Le serveur redirige les messages à tous les clients en JSON

## PREMIÈRE TECHNOLOGIE SÉLECTIONNÉE (LA NOUVELLE)

**Technologie :** serveur Node.JS

**URL :** <https://www.colyseus.io>

## JUSTIFIER VOTRE CHOIX TECHNOLOGIQUE POUR CETTE TECHNOLOGIE

**Expliquer à l'aide d'une argumentation rationnelle votre choix technologique. Établir votre justification à l'aide de liens avec les fonctionnalités, contraintes et risques de votre projet. Un tableau comparatif permettant de synthétiser votre réflexion pourrait être un apport judicieux à vos explications.**

J'ai déjà utilisé du Javascript et du Typescript ce qui facilitera la prise en main de Node.JS.

Je n'ai cependant pas utilisé de serveur Node.JS.

Grille de comparaison(Serveur Node.JS/Serveur Apache) :

<https://docs.google.com/presentation/d/1W4HzrMWQYBOEFAWyfxzaN91jq76qxQlqh94lqEMMRcU/edit?usp=sharing>

J'ai choisi d'utiliser le framework Colyseus.io qui simplifie beaucoup le multijoueur grâce au concept de Room. On peut trouver plusieurs tutoriels pour Colyseus.io.

## DEUXIÈME TECHNOLOGIE SÉLECTIONNÉE (LA CONNUE)

Technologie : Unity

URL : <https://unity.com/fr>

## JUSTIFIER VOTRE CHOIX TECHNOLOGIQUE POUR CETTE TECHNOLOGIE

**Expliquer à l'aide d'une argumentation rationnelle votre choix technologique. Établir votre justification à l'aide de liens avec les fonctionnalités, contraintes et risques de votre projet. Un tableau comparatif permettant de synthétiser votre réflexion pourrait être un apport judicieux à vos explications.**

Je n'ai jamais utilisé Unity sauf lors des cours de réalité virtuelle donc mon apprentissage de Unity est récent.

Unity a l'air assez simple à prendre en main.

La syntaxe C# ressemble beaucoup à celle de Java. Cela facilitera donc beaucoup mon apprentissage car j'ai beaucoup utilisé Java lors de mes études.

L'avantage que j'ai à utiliser Unity par rapport à d'autres moteurs de jeu est que j'aurais donc plus de temps pour le prendre en main. Unreal Engine utilise le C++ avec lequel je ne suis pas familier.