

## Rapport Projet Annuel

**Rédigé par :**

*Guillaume Morel, Nathan Panchout, Anthony Zago,  
Mathieu Michel, Hugo Sadaune, Pierre Sinegre.*

## **Sommaire :**

<b>Introduction :</b>	<b>3</b>
<b>I/ Description fonctionnelle</b>	<b>4</b>
<b>II/ Description technique</b>	<b>7</b>
<b>III/ Bilan des tests</b>	<b>10</b>
<b>IV/ Bilan moral</b>	<b>10</b>
<b>V/ Problèmes</b>	<b>11</b>

## Introduction :

La plateforme AGORA est une plateforme de jeu de société multijoueurs en ligne.

La version actuellement en ligne possède 8 jeux.

Le projet AGORA est un ancien projet, sur lequel sont passées plusieurs équipes au fil des années, malheureusement le code fourni par ces équipes n'a pas toujours été de qualité, de ce fait l'architecture est bancal et comme le code n'a pas du tout été documenté, il s'est avéré compliqué de reprendre le projet existant.

Au départ, pour ce projet, nous devions ajouter 3 jeux et refaire le design de la plateforme en ajoutant quelques fonctionnalités, nous avons même fait les documents de gestion de projets pour ces jeux (STB).

Mais, Il était donc impossible, pour nous, de reprendre cette plateforme accumulant des défauts dans le code.

Nous avons donc, au cours d'une réunion avec notre client, proposé de refaire la plateforme de zéro, avec une architecture propre respectant le modèle MVC mais n'implémentant que 2 jeux.

Le but de notre nouvelle architecture est qu'elle soit extensible et bien documentée, utilisant des technologies accessibles et faciles à apprendre, afin que les équipes après nous ne tombent pas sur les mêmes problèmes que nous avons rencontrés.

# I/Description fonctionnelle

Ici, nous allons décrire toutes les fonctionnalités de notre application.

## 1)La plateforme

### a) La page principale

Ceci est la page principale d'agora :

[Screenshot page principale]

Cette page permet d'accéder à toutes les fonctionnalités de notre application.

On peut sur cette page :

- Accéder à l'inscription sur la plateforme agora avec confirmation par mail
- Se connecter
- Si on est connecté, Voir son profil, le modifier ou se déconnecter
- Participer à une discussion sur le tchat
- Accéder au module permettant de rejoindre une partie
- Accéder au module permettant de créer une partie pour un jeu
- Accéder au classement des joueurs
- Accéder à la page permettant de contacter les développeurs
- Accéder à la page de modération si vous êtes modérateur
- Accéder à la plateforme d'administration d'AGORA si vous êtes administrateur

Le tchat est disponible sur toutes les pages de la plateforme. Pour pouvoir discuter, il faut être connecté.

Ce tchat conserve l'historique des 20 derniers messages seulement, par soucis de limitation des capacités du serveur de l'université de Rouen-Normandie.

### b) Gestion des utilisateurs/joueurs

Voici la page permettant de s'inscrire sur AGORA :

[Screenshot inscription]

Pour s'inscrire, on a besoin d'indiquer un email valide, car nous envoyons une demande de confirmation.

On peut par la suite changer son mot de passe sur la page de profil.

Exemple de confirmation par mail :

Bienvenue agoraTest !



Boîte de réception x



**Plateforme Agora** <agora.dev.test@gmail.com>

À moi ▾

Bonjour agoraTest !

Pour valider votre compte utilisateur, merci de vous rendre sur <http://localhost:8000/register/confirm/5twl1349-hhWzA1HtF9QCQvqs3zkHV05JRSXj-qG9ck>

Ce lien ne peut être utilisé qu'une seule fois pour valider votre compte.

Cordialement,  
L'équipe

Voici la page de profil d'un utilisateur connecté :

Vos informations

### Modifier les infos de votre compte

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum sem libero, congue eu lacinia id, accumsan vitae enim. Pellentesque porta purus elit, eget consequat lacus consectetur id.

---

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum sem libero, congue eu lacinia id, accumsan vitae enim. Pellentesque porta purus elit, eget consequat lacus consectetur id.

### Vos informations

Modifier

### A propos

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum sem libero, congue eu lacinia id, accumsan vitae enim. Pellentesque porta purus elit, eget consequat lacus consectetur id.

Vous pouvez ici modifier votre profil et accéder aux informations sur votre compte.

Sur notre plateforme, un utilisateur peut posséder un des trois rôles suivant :

- utilisateur, le rôle par défaut
- modérateur, il peut gérer les parties sur la plateforme
- administrateur, il peut gérer les parties et les utilisateurs inscrits sur la plateforme

La gestion des utilisateurs se fait sur la plateforme administrateur que nous allons décrire plus tard.

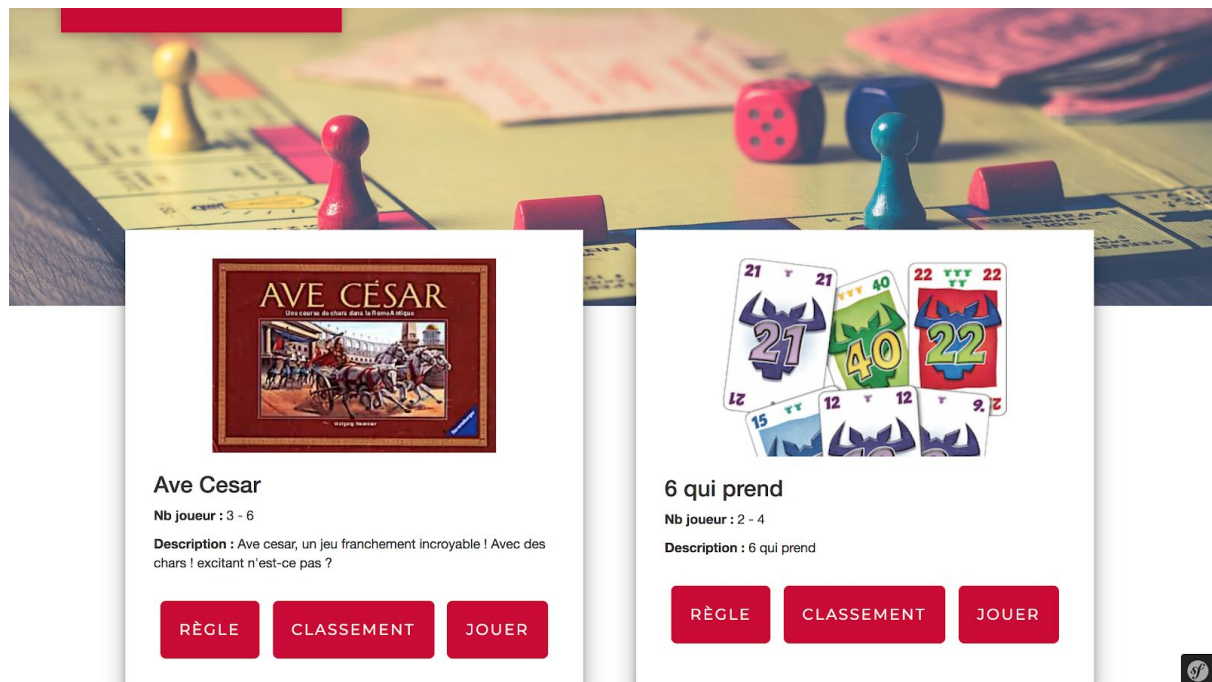
## c) Liste des jeux / Création de parties

Voici la page permettant de créer une partie après avoir sélectionné un jeu :

Ici, on peut définir les paramètres de la partie qu'on veut créer tels que :

- Le nom du lobby
- Le nombre de joueurs
- Définir la partie comme étant privée
- Si la partie est privée, indiquer un mot de passe

Les utilisateurs d'AGORA peuvent rejoindre la partie qu'on vient de créer sur la page prévue à cet effet.



## d) Joindre une partie

Voici la page permettant de rejoindre une partie :

On peut ici trier les parties par nom de lobby, nombre de joueurs présents, date de création, etc... pour chaque jeu.

Pour rejoindre, il suffit de cliquer sur le champ "rejoindre" de la ligne de la partie concernée.

Si la partie est privée, on demande à l'utilisateur d'entrer le mot de passe du lobby.

Il est possible de revenir dans une partie qu'on a déjà commencée (si l'onglet du jeu a été fermé par exemple) grâce à l'onglet "Mes parties".

## e) La plateforme administrateur

Cette page n'est accessible que par l'administrateur de la plateforme.

Ici, il peut :

- créer des comptes à la chaîne pour les démonstrations lors des événements universitaires.
- Supprimer des utilisateurs
- Ajouter/supprimer des rôles aux utilisateurs (sauf le rôle administrateur, qui est unique)

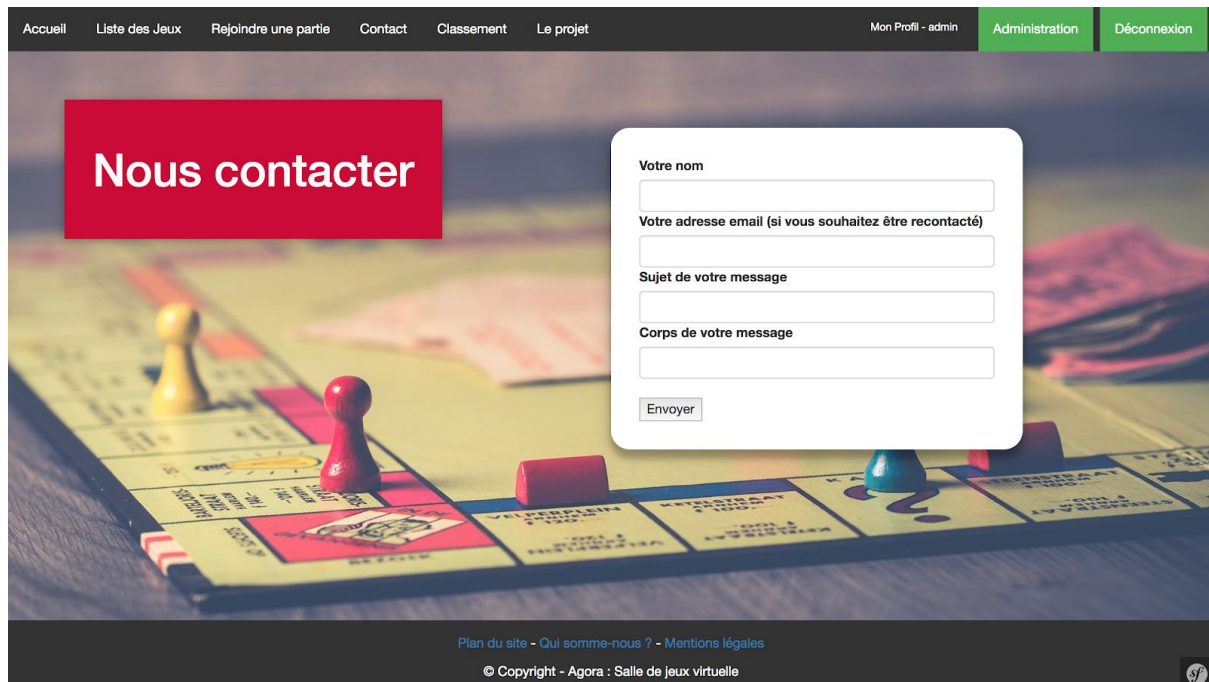
## f) La plateforme de modération

Cette page n'est accessible que par les modérateurs et l'administrateur

Ici les modérateurs et l'administrateur peuvent supprimer des parties, générer des utilisateurs provisoires et les administrer

## g) Contact avec les développeurs

Voici la page permettant d'envoyer un message aux développeurs :



Accueil Liste des Jeux Rejoindre une partie Contact Classement Le projet Mon Profil - admin Administration Déconnexion

# Nous contacter

Votre nom

Votre adresse email (si vous souhaitez être recontacté)

Sujet de votre message

Corps de votre message

Envoyer

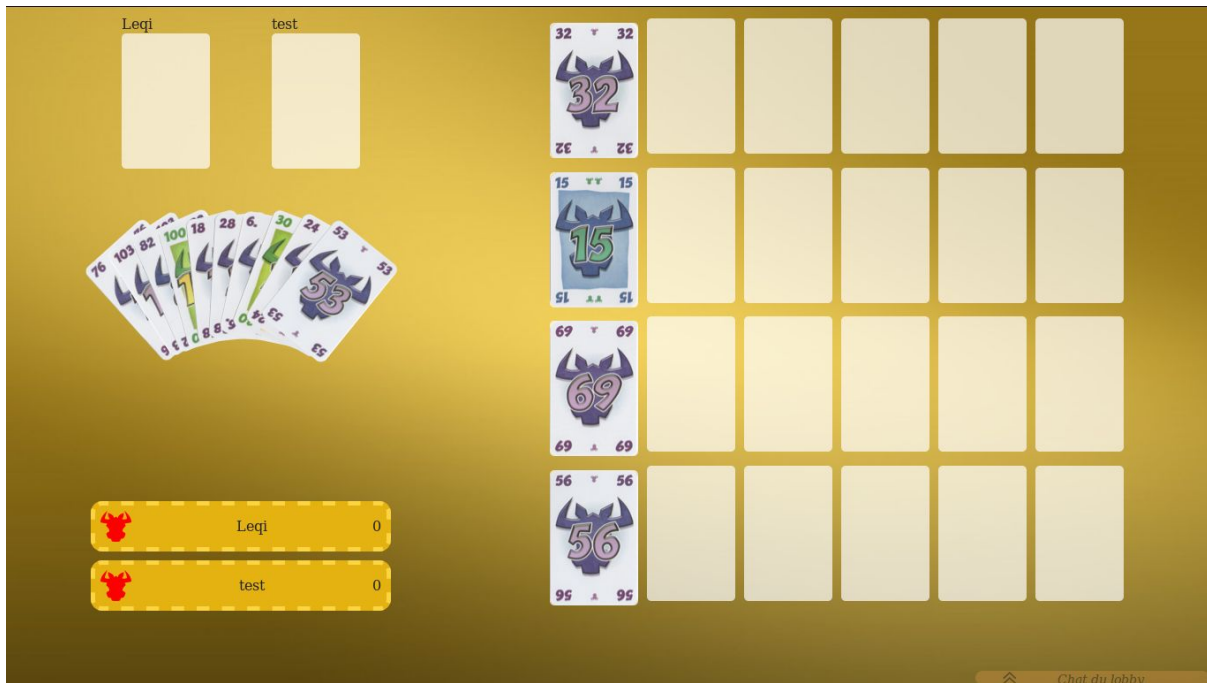
Plan du site - Qui sommes-nous ? - Mentions légales

© Copyright - Agora : Salle de jeux virtuelle

Cette page sert à envoyer des remarques aux développeurs sur des bugs découverts, des améliorations possibles de la plateforme, etc ...

## 2) Le 6 qui prend

Voici la vue du 6 qui prend :



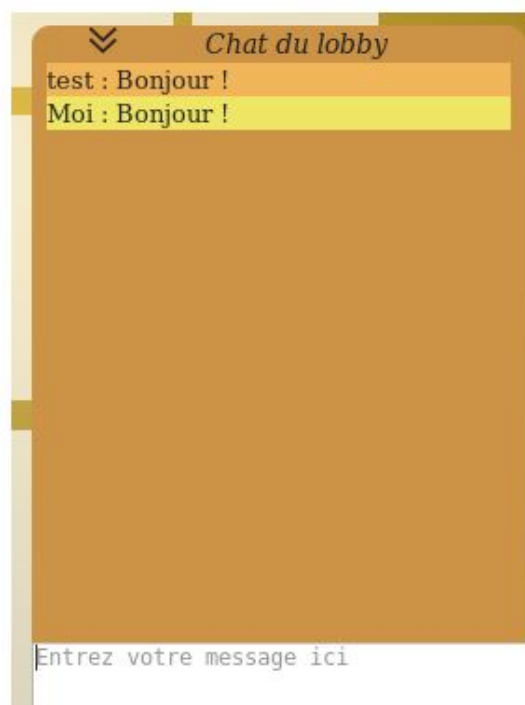
Les règles du 6 qui prend sont disponibles à l'adresse suivante :

<https://www.gigamic.com/files/catalog/products/rules/rules-6quiprend-05-2012.pdf>

Pour déposer une carte, il suffit de cliquer dessus et d'attendre que les autres joueurs aient fait leur choix. Le client de ce jeu a été conçu pour qu'il soit intuitif et facile à utiliser.

La différence avec l'ancien 6 qui prend est que nous avons fait en sorte que la vue tienne sur une page pour le confort de jeu.

Nous avons aussi ajouté un tchat permettant aux membres de la partie de communiquer.

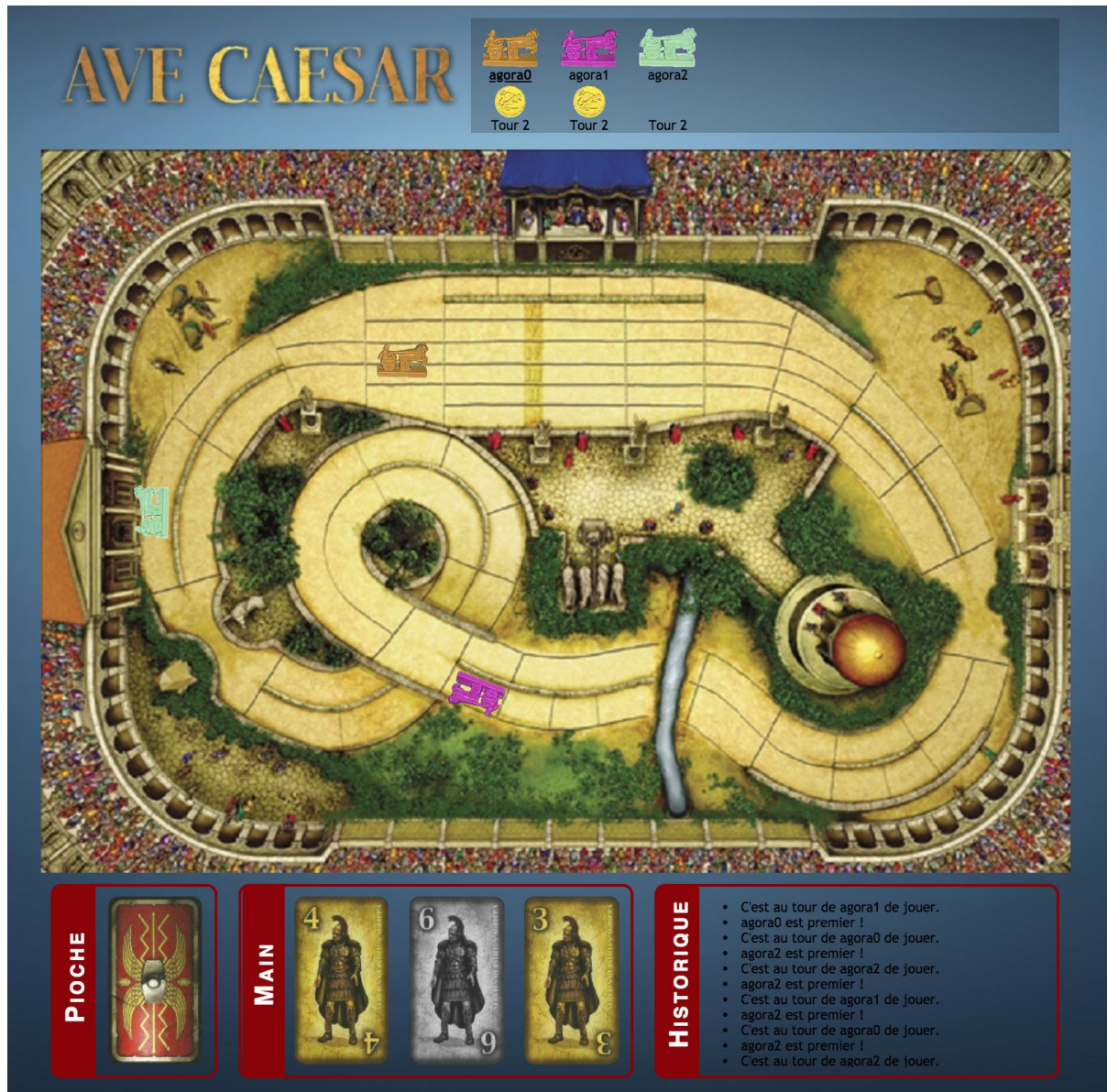




Pour écrire dans le tchat ou le voir, il suffit de cliquer sur la petite flèche à gauche de la barre sur laquelle il est marqué "Chat du lobby" pour le dérouler. Nous avons ajouté une transparence quand l'utilisateur n'a pas sa souris sur le tchat afin qu'il puisse voir les cartes du plateau sans problème. Ce tchat, comme celui de la page principale, conserve dans un historique les 20 derniers messages.

### 3) Ave Cesar

Voici la vue d'Ave Cesar :



Les règles de l'Ave Cesar sont disponibles à l'adresse suivante :

[http://jeuxstrategie.free.fr/Ave\\_cesar\\_complet.php](http://jeuxstrategie.free.fr/Ave_cesar_complet.php)

Chaque joueur dispose d'une main de 3 cartes. Les cartes grisées ne sont pas jouables dans la configuration actuelle du jeu. Pour jouer son tour il suffit de sélectionner une carte

jouable et de cliquer sur la plateau (voir règle pour les déplacements valides). Si le joueur ne possède pas de carte lui permettant de jouer il lui est offert la possibilité de passer son tour.

Lorsque le joueur effectue l'action "*Déplacement*" ou "*Passer*" c'est au joueur suivant de jouer. Sachant qu'une carte de son deck lui est distribuée automatiquement si il en a utilisé une.

Un historique des actions marquantes de la partie est présent en bas à droite de l'écran. On peut voir en haut de l'écran la liste des joueurs ainsi que pour chacun d'eux leur pion associé, leur tour et une pièce dorée présente si le joueur est passé dans l'allée impériale.

## 4) Installation

Nous avons créé un script “install.sh” permettant d’installer notre application facilement.

Ce script installe les bundles nécessaires au fonctionnement de l’application ainsi que les données à insérer dans la base de données dont notre application a besoin.

Ce script insère aussi le compte administrateur , de nom d’utilisateur “admin” et de mot de passe “admin” (à changer absolument bien sûr !) nécessaire pour pouvoir administrer l’application car la plateforme ne peut posséder qu’un seul administrateur par défaut (il est possible d’en avoir plusieurs si besoin).

## II/Description technique

### 1) Technologies utilisées

Le choix des technologies que nous avons utilisées pour réaliser notre application est important. Ce choix doit être cohérent avec les besoins du client et ces technologies doivent être durables pour que les prochaines équipes de développement de la plateforme puissent continuer à la faire évoluer sans avoir à changer de technologies.

Nous allons donc dans cette partie justifier nos choix.

#### a) Symfony

Pour la plateforme AGORA, nous avons choisi d’utiliser le framework Symfony.

Symfony est un framework PHP facilitant le développement d’application web.

Nous avons choisi d’utiliser Symfony car :

- Symfony possède une belle documentation, une communauté active ainsi qu’une multitude de tutoriels, facilitant l’apprentissage pour ceux n’ayant jamais utilisé ce framework, voire jamais utilisé de framework du tout (C’était le cas pour la plupart des membres de notre équipe).
- Symfony possède énormément de modules (bundles) qui nous facilite grandement le développement, comme par Exemple FOSUserBundle qui permet la gestion des utilisateur, RatchetCBoden qui gère les websockets de notre plateforme , très utile pour les jeux et les tchats.
- Symfony propose un module permettant d’effectuer des tests logiciels facilement.
- Symfony est beaucoup utilisé dans les entreprises.
- Symfony est français .

#### b) Websockets

Les websockets fonctionnent grâce au bundle RatchetCBoden, il nous permet de créer un socket auquel un client javascript peut se connecter puis d’effectuer des traitements lorsqu’un utilisateur :

- Se connecte

- Envoie un message
- Se déconnecte (ou qu'il perd la connexion)
- En cas d'erreur

La plupart de ces traitements se terminent par l'envoi de messages aux utilisateurs (en sauvegardant leur connexion au préalable).

Ces fonctionnalités sont indispensables pour le développement des jeux car elles nous permettent de gérer la sauvegarde des données, mais aussi le multi-joueurs dans nos jeux. Les interactions entre le client et le serveur dans un jeu se font exclusivement via ces socket.

On différencie les demandes des clients grâce au champ "type" du paquet envoyé. Par exemple, si un client envoie un message dans le tchat, il envoie un objet Javascript, encodé en JSON, qui comporte un champ type, qui ici aura pour valeur 'message', indiquant à la socket qu'elle a reçu un message d'un client destiné au tchat.

Nous pensions dans un premier temps utiliser NodeJs (*influencé par le groupe travaillant sur AGORA l'année dernière*), mais nous avons découvert que Symfony propose un bundle implémentant les websockets, RatchetCBoden, nous avons donc décidé d'utiliser celui-ci car il permet d'utiliser directement des services créés dans le projet Symfony depuis le serveur de socket.

### c) Doctrine

Doctrine nous permet d'accéder à notre base de données et de la modifier facilement en insérant et en récupérant directement des objets PHP instanciés.

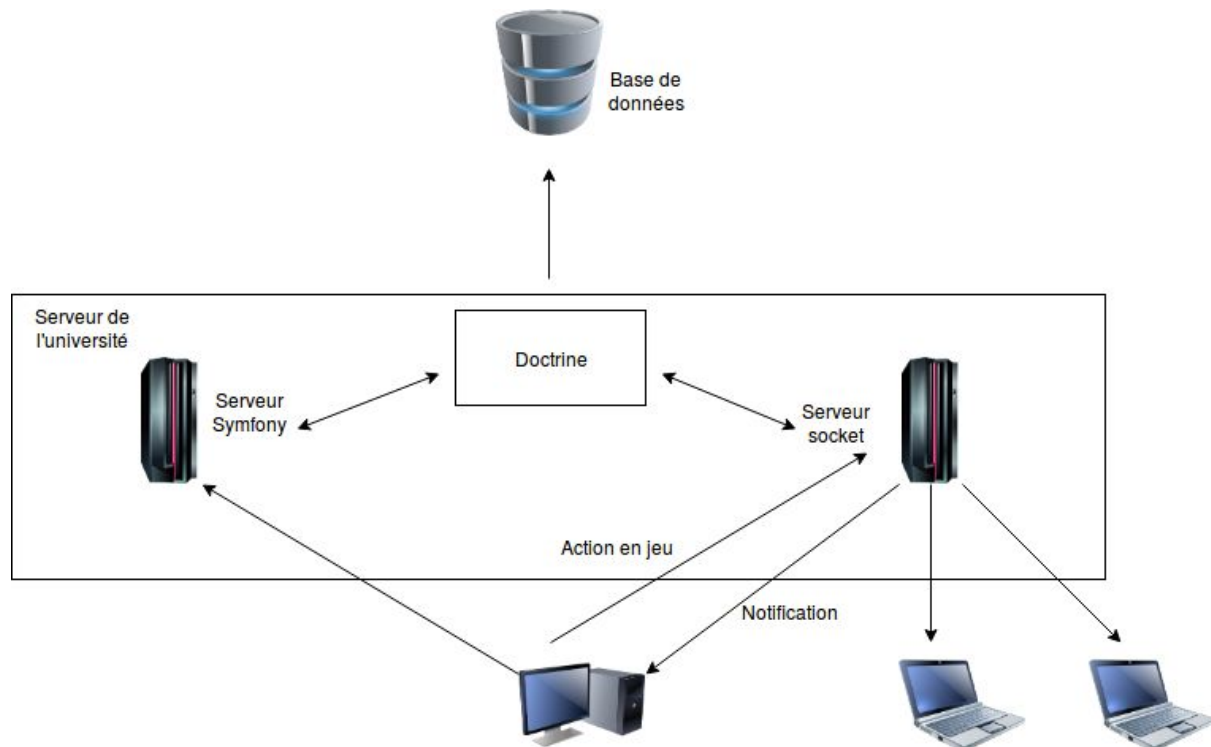
Cela nous évite d'avoir à rédiger des requêtes SQL complexes et donc d'éviter les erreurs de syntaxe SQL.

De plus, Doctrine protège notre application des injections de code (SQL et javascript).

## 2) Architecture

Dans cette partie, nous allons expliquer comment est formée l'architecture de la plateforme et des jeux.

Voici l'architecture générale de la plateforme :



Sur ce diagramme, on peut voir que le client accède aux pages web par Symfony, mais aussi les échanges entre le serveur de socket et les clients. Doctrine nous permet d'accéder à la base de données plus simplement à partir de Symfony et des sockets.

### 3) Algorithmes utilisés / Boucles de jeu

Pour que nos jeux multijoueurs fonctionnent sur notre plateforme, nous utilisons les websockets.

Le déroulement d'une partie de jeu se constitue d'échanges de messages (contenant des données nécessaires au déroulement de l'action que doit effectuer celui qui reçoit le message).

Pour que le serveur ou le client sache quoi faire en recevant un message, nous définissons un champ "type" à l'intérieur du message qui est en fait un objet JSON que nous transformons en chaîne de caractères.

Chaque action modifie l'état du jeu côté serveur et en informe les autres clients (*joueurs*) que l'action a eu lieu afin de garder un état synchronisé entre les clients et le serveur. L'état complet du jeu est envoyé entièrement uniquement lors de la connexion/reconnexion à une partie (i.e chargement/rechargement d'une page de jeu).

## a) 6 Qui Prend

Pour que la partie démarre, il faut attendre que le nombre requis de joueurs pour commencer la partie soit atteint, le socket envoie donc un message de type “begin” indiquant à l’hôte qu’il peut démarrer la partie.

Une fois la partie démarrée, la boucle de jeu commence. Cette boucle consiste en des échanges de messages entre les clients et le socket. Elle se déroule de la manière suivante :

- Les joueurs posent une carte en cliquant sur une carte de leur main, chaque client envoie ainsi un message de type “readyCard”.
- Quand un joueur pose une carte, le serveur envoie un message de type “readyCard” à tous les clients pour qu’ils affichent bien qu’un joueur a sélectionné une carte.
- Quand tous les joueurs ont posé leurs cartes , le serveur définit l’ordre de passage des joueurs et envoie un message de type “READY”.
- Les clients retournent donc les cartes posées par les joueurs puis attendent leurs tours.
- Quand c’est au tour d’un joueur, il envoie sa carte qu’il veut poser s’il peut la poser sans prendre de ligne, sinon il choisit où la poser puis l’envoie dans un message de type “card”.
- Le serveur vérifie que le coup est valide puis indique quel joueur doit jouer, modifie le score si besoin et envoie un message de type “refreshBoard” à tous les joueurs pour rafraîchir le plateau et un message de type “refreshHand” au joueur qui a posé sa carte.
- Si plus personne n’a de carte et qu’aucun joueur n’a un score supérieur ou égal à 66, le serveur refait le plateau de jeu, redistribue les cartes puis envoie un message de type “refreshAll” à tous les joueurs.
- Si plus personne n’a de carte et qu’un joueur à un score supérieur ou égal à 66, alors le serveur envoie un message de type “END” aux clients, modifie le classement ELO des joueurs puis supprime la partie de la base de données.
- Le client affiche le résultat de la partie.

## b) Ave Cesar

Pour que la partie démarre, il faut attendre que le nombre requis de joueurs pour commencer la partie soit atteint, le serveur de socket envoie donc un message de type “start” indiquant à tous les joueurs que la partie commence.

Une fois la partie démarrée, la boucle de jeu commence. Cette boucle consiste en un échange de messages entre les clients et le serveur de socket. Ce serveur s’occupe de recevoir les actions et de vérifier qu’elles sont valides avant d’en enregistrer l’effet dans la base de données et d’informer les autres clients des modifications. Elle se déroule de la manière suivante :



- Le premier joueur est le créateur de la partie, il joue son tour (sélectionne une carte et clique sur une case du plateau). Le client envoie un message "move" au serveur avec les informations de déplacement (*carte sélectionnée et position ciblée*).
- Le serveur valide ou non l'action, si elle n'est pas valide le serveur informe l'émetteur de la non validité de l'action. En revanche si elle est valide le serveur modifie l'état du jeu dans la base de donnée et informe les joueurs des changements apportés (par exemple : la nouvelle position du joueur, le prochain joueur qui doit jouer, et si le joueur a terminé la course).
- Les autres joueurs reçoivent ces informations sous la forme d'un objet javascript de type "move" et modifient leur vue grâce aux informations fournies pour la faire correspondre à l'état du jeu.
- Si tous les joueurs ont fini les 3 tours, alors le serveur envoie un message de type "finish" avec comme contenu le classement de la partie.
- Le client se charge d'afficher ce classement.

### c) Algorithme d'implémentation de l'ELO

Pour implémenter cet algorithme il faut tout d'abord classer les joueurs à la fin de la partie. Ensuite pour chaque joueur, on applique la formule mathématique de modification de l'ELO pour un jeu à 2 joueurs (notamment utilisé dans le cas du jeu d'échecs). Avec tous les autres joueurs de la partie, si le joueur est devant au classement il gagne des points, si il est derrière il en perd. le montant de points gagnés ou perdus dépend évidemment des ELO des deux joueurs, cela grâce un système de paliers que l'on a mis en place afin de moduler les gains de points.

## III/Bilan des tests

### 1) Tests unitaires

Les tests unitaires ont été effectués grâce à Symfony.

La plupart des tests ont été effectués sur les jeux. Nous testons les modifications de la base de données et les données fournies par le serveur de socket au client. Ainsi, nous nous sommes assurés que les règles des jeux soient respectées (Distribution des cartes correctes, pas d'erreur dans les scores, etc...).

Toutes les fonctionnalités qui ont été testées ont validé les tests.

### 2) Tests fonctionnels

A chaque étape de développement, nous avons testé les fonctionnalités ajoutées à la plateforme en les utilisant intensivement.

Nous avons testé principalement les événements se déclenchant lors de l'action d'un utilisateur.

Par exemple, pour les jeux, nous avons vérifié que toutes les actions possibles se déroulaient comme prévu, qu'aucun comportement non désiré n'ait lieu.

## IV/Bilan moral

### 1) Ce que nous a apporté ce projet

Nous avons appris à nous organiser en équipe: à nous répartir les tâches équitablement, à gérer les conflits et à nous réorganiser en cas d'abandon d'un membre de l'équipe.

Nous avons appris à utiliser de nouvelles technologies utilisées dans de nombreuses entreprises comme le framework Symfony et les websockets.

Nous avons appris à réaliser les documents nécessaires au bon déroulement du projet.

## V/ Problèmes

### 1) Risques avérés

Nous avons analysé les risques possibles au cours de notre projet et les solutions à apporter.

Deux des risques se sont déclenchés :

- Abandon de membres de l'équipe (2 membres sont partis), causant un manque d'effectif pour accomplir les tâches initialement prévues.
- Charge de travail en cours importante, mettant en suspens le développement de la plateforme.

Nous avons mis en place plusieurs solutions :

- Prévoir de la marge sur le temps nécessaire pour accomplir nos tâches.
- Revoir les objectifs avec le client afin de réduire la quantité de travail.

Ces solutions ont été mises en place afin d'éviter d'avoir du retard dans le développement de notre plateforme.

### 2) Difficultés rencontrées

Nous avons rencontré quelques difficultés lors du développement de la plateforme qui sont les suivantes :

- Reprendre le code après 3 semaines de pause pour travailler les cours et les autres projets. Pour palier à ce problème, nous nous sommes assuré de bien commenter et documenter le code avant de passer à autre chose pour que la reprise soit plus facile et que l'on ne perde pas de temps.



- Appréhension de Symfony, nous ne connaissions pas Symfony avant ce projet, nous avons donc dû apprendre à utiliser ce framework. Nous avons suivi le tutoriel d'OpenClassroom.

### 3) Améliorations possibles

Par manque de temps et à cause des risques rencontrés, certaines fonctionnalités (moins prioritaires) n'ont pas été implémentées :

- La traduction de la plateforme.
- Les invitations à rejoindre une partie.

Ces fonctionnalités restent donc des idées pour les futures versions, nous nous sommes assurés que ces fonctionnalités puissent être implémentées dans le futur.

Le client a également émis l'idée de nouvelles fonctionnalités (après l'analyse des besoins) pour les futures versions :

- Permettre le remplacement d'un joueur en jeu.
- Pouvoir accéder à un profil en cliquant sur le pseudo d'un joueur (et donc la création de pages de profils plus détaillées).
- Intégrer la documentation directement sur la plateforme.