Magix API v1.0.8

Objet du document

Ce document contient toutes les informations afin d'être en mesure de se connecter au serveur du jeu Magix.

Table des matières

Objet du document 1
Table des matières 1
Le jeu Magix 2
Page d'authentification 2
Page « lobby » 3
Page de jeu 3
L'enregistrement 4
Créer son « Deck » 4
Se connecter à l'API 5
La boîte de chat (clavardage) 6
Styliser votre boîte de chat 6
Les services disponibles (l'API) 7
Connexion au serveur 7
Déconnexion du serveur 7
Créer/Joindre une partie 8
Pour faire une action (jouer une carte, terminer son tour, attaquer) 9
L'état de la partie en cours 10
Le jeu 11
L'objectif 11
Les cartes 11
Temps accordé pour jouer son tour 11
Autres notes importantes 12
Éviter d'être banni du serveur et les délais d'appel 12
Le quasi temps-réel 12
Distinguer un message texte d'un objet dans une variable 12

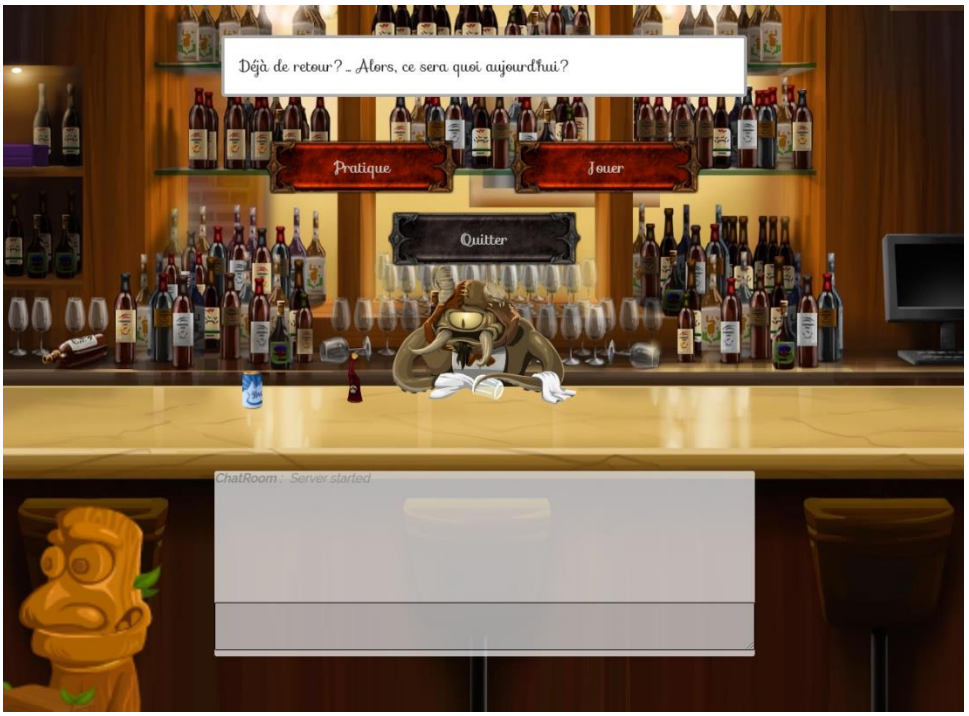
Le jeu Magix

Magix est un serveur d'un jeu de cartes basé sur la technologie PHP, HTML, Node, JavaScript, MongoDB et AJAX. Celui-ci peut être joué contre l'ordinateur ou un autre joueur.

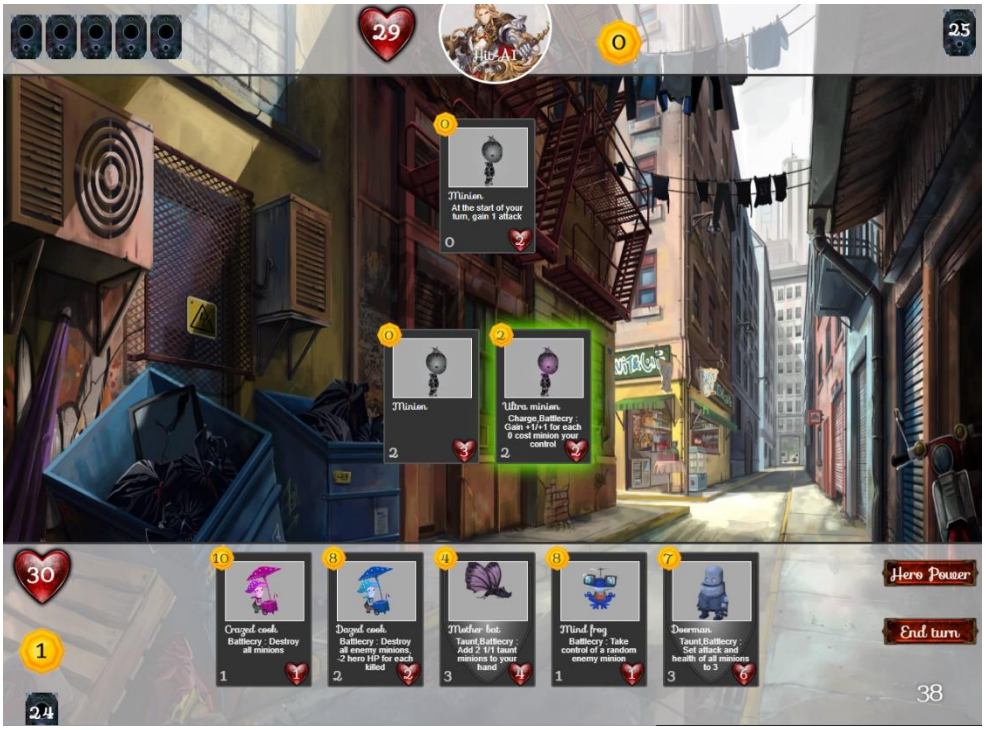
L'API, qui est disponible aux étudiants, permet de concevoir un jeu avec une interface graphique au choix. Voici un exemple d'interface.

Page d'authentification

Page « lobby »



Page de jeu



Le commencement

Avant même de se connecter à l'API, l'étudiant doit avoir terminé de créer son compte sur le serveur Magix.

L'enregistrement

Pour ce faire, il doit aller au lien suivant, et remplir le formulaire d'inscription :

https://magix.apps-de-cours.com/server/#/signup

Créer son « Deck »

Un deck est un ensemble de 30 cartes sélectionnés par l'utilisateur. Il faut donc se connecter sur le site Web, puis aller dans la section « Deck » pour créer son deck.

Lorsque votre deck est terminé, vous pouvez faire votre version du jeu.

Se connecter à l'API

Pour créer un jeu, il est nécessaire de communiquer avec le serveur de Magix. Vous devrez donc faire appel à son API.

Voici un exemple d'appel qui permet de faire l'opération « signin ». Il est **FORTEMENT** conseillé d'utiliser cette fonction tel quel et de la placer dans « CommonAction ».

```
/**
 * data = array('key1' => 'value1', 'key2' => 'value2');
 */
public function callAPI($service, array $data) {
        $apiURL = "https://magix.apps-de-cours.com/api/" . $service;

        $options = array(
            'http' => array(
                'header'  => "Content-type: application/x-www-form-urlencoded\r\n",
                'method'  => 'POST',
                'content' => http_build_query($data)
            )
        );
        $context  = stream_context_create($options);
        $result = file_get_contents($apiURL, false, $context);

        if (strpos($result, "<br") !== false) {
                var_dump($result);
                exit;
        }

        return json_decode($result);
}
```

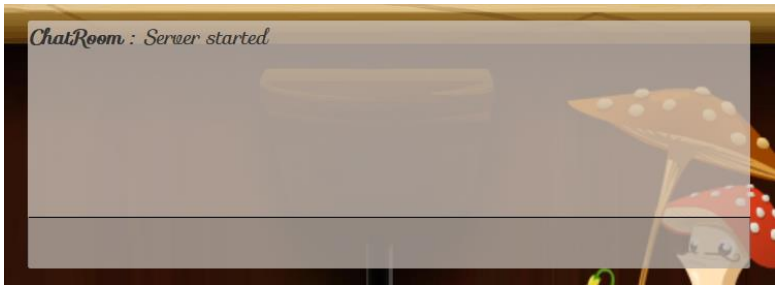
Exemple d'appel (à intégrer dans votre action)

```
$data = [];
$data["username"] = "Falcor";
$data["password"] = "AAAaaa111";

$result = parent::callAPI("signin", $data);

if ($result == "INVALID_USERNAME_PASSWORD") {
        // err
}
else {
        // Pour voir les informations retournées : var_dump($result);exit;
        $key = $result->key;
}
```

La boîte de chat (clavardage)



Pour faire apparaitre la boîte de *chat* dans votre page Web, il s'agit d'injecter votre clé de session (reçu lors de l'authentification) dans l'extrait de code HTML suivant :

```
<iframe style="width:700px;height:240px;"
        src="https://magix.apps-de-cours.com/server/#/chat/votre-clé-ici">
</iframe>
```

Styliser votre boîte de chat

Afin de modifier le style de votre boîte de chat, il s'agit de faire ce qui suit.

1- Dans la création du iframe, ajouter l'événement *onload*.

```
<iframe style="width:700px;height:240px;" onload="applyStyles(this)"
        src="https://magix.apps-de-cours.com/server/#/chat/votre-clé-ici">
</iframe>
```

2- Dans votre JavaScript, déclarer la fonction « applyStyles() ». Voici un exemple :

```
const applyStyles = iframe => {
        let styles = {
                fontColor : "#333",
                backgroundColor : "rgba(87, 41, 5, 0.2)",
                fontGoogleName : "Sofia",
                fontSize : "20px",
        }

        iframe.contentWindow.postMessage(JSON.stringify(styles), "*");
}
```

Notes :

- Si vous désirez modifier la police de caractères, utilisez « fontGoogleName ». Cela doit correspondre au nom d'un font disponible à cet endroit : https://fonts.google.com/
- Il n'y a que les styles plus haut qui sont acceptés.

Les services disponibles (l'API)

Connexion au serveur

Pour faire une authentification auprès du serveur, il faut avoir complété son inscription (voir plus haut). Lors d'une connexion, le serveur vous retournera une clé. Celle-ci doit être conservée en session car tous les autres appels au serveur en dépendent. Cette clé permet au serveur de « vous reconnaitre » (savoir qui vous êtes).

Nom du service	signin	
Paramètres	username password	Le nom de votre personnage Votre mot de passe
Retour (succès)	Informations de votre personnage et la clé de session	La clé aura cette forme (50 car.): « afsc9sflasmknc5lkntasd9yhcbasdfnasd9fcn »
Retour (erreur)	"INVALID_USERNAME_PASSWORD"	- Erreur d'authentification

Pour savoir comment implémenter ce service, veuillez regarder l'exemple plus haut.

Déconnexion du serveur

Lorsque vous êtes connectés, vous pouvez vous déconnecter en passant au service votre clef de session.

Nom du service	signout	
Paramètres	key	Votre clé *doit être envoyée dans un tableau, même s'il n'y a que la clé en paramètre. Exemple :* *$data = [];* *$data["key"] = "asfsdafdsf54879y…";*
Retour (succès)	"SIGNED_OUT"	
Retour (erreur)	" INVALID_KEY"	

Créer/Joindre une partie

Permet de jouer une partie contre une autre personne (pvp), ou contre l'ordinateur (training).

Nom du service	games/auto-match	
Paramètres	key type private-key (optionnel)	Votre clé "PVP" ou "TRAINING" Permet de créer une partie privée entre 2 joueurs, en utilisant un mot de passe pour accéder à la partie
Retour (succès)	"JOINED_PVP" "CREATED_PVP" "JOINED_TRAINING"	
Retour (erreur)	" INVALID_KEY" "INVALID_GAME_TYPE" " DECK_INCOMPLETE" "MAX_DEATH_THRESHOLD_REACHED"	 Lors de tournoi seulement. Lorsque vous avez cette erreur, vous êtes mort!

Après cet appel, vous êtes dans une partie, *game on!*

Pour faire une action (jouer une carte, terminer son tour, attaquer)

Nom du service	games/action	
Paramètres	key	Votre clé
	type	"END_TURN"
	ou	
	type	"HERO_POWER"
	ou	
	type	"PLAY" et
	uid	ex : 23 (identifiant unique de la carte lors de la partie)
	ou	
	type	"ATTACK"
	uid	uid de la carte
	targetuid	uid de la carte attaquée ou 0 (zéro) pour le héros adverse
Retour (succès)	L'état de la partie, voir page suivante	
Retour (erreur)	"INVALID_KEY"	
	"INVALID_ACTION"	Action invalide
	"ACTION_IS_NOT_AN_OBJECT"	Mauvaise structure de données
	"NOT_ENOUGH_ENERGY"	La carte coûte trop cher à jouer
	"BOARD_IS_FULL "	Pas assez de place pour la carte
	"CARD_NOT_IN_HAND"	La carte n'est pas dans votre main
	"CARD_IS_SLEEPING"	Carte ne peut être jouée ce tour-ci
	"MUST_ATTACK_TAUNT_FIRST"	Une carte taunt empêche ce coup
	"OPPONENT_CARD_NOT_FOUND"	La carte attaquée n'est pas présente sur le jeu
	"CARD_NOT_FOUND"	La carte cherchée (uid) n'est pas présente
	"ERROR_PROCESSING_ACTION"	Erreur interne, ne devrait pas se produire
	"INTERNAL_ACTION_ERROR"	Autre erreur interne, ne devrait pas se produire
	"HERO_POWER_ALREADY_USED"	Pouvoir déjà utilisé pour ce tour

L'état de la partie en cours

Lorsque le personnage est dans la partie, il faut demander régulièrement son état afin de savoir si votre vie a diminué, quelles sont les cartes dans votre main, etc.

Il y doit y avoir un délai de 1 seconde minimum entre chaque appel. Autrement, l'appel pourrait être refusé et vous pourriez même être déconnecté!

Nom du service	games/state	
Paramètres	Key	Votre clé
Retour (succès)	"WAITING" ou "LAST_GAME_WON" ou " LAST_GAME_LOST" ou Un document JSON contenant l'état de la partie (voir exemple plus bas)	- En attente d'un autre joueur - La partie n'existe plus, mais la dernière partie jouée a été gagnée - La dernière partie n'existe plus et vous l'aviez perdue.
Retour (erreur)	"INVALID_KEY"	

Exemple de résultat (JSON)


```
{
	"remainingTurnTime":24,
	"yourTurn":true,
	"heroPowerAlreadyUsed" : false,
	"hp":30,
	"mp":0,
	"maxMp":1,
	"hand":[
		{"id":4,"cost":2,"hp":3,"atk":2,"mechanics":[], "uid":3,"baseHP":3},
		{"id":22,"cost":7,"hp":7,"atk":7,"mechanics":[],"uid":5,"baseHP":7},
		{"id":10,"cost":3,"hp":3,"atk":3,"mechanics":["taunt", "charge"],"uid":6,"baseHP":3}
	],
	"board":[
		{"id":2,"cost":1,"hp":1,"atk":2,"mechanics":[],"uid":7,"baseHP":1,"state":"SLEEP"}
	],
	"welcomeText" : "My life for Aiur!",
	"heroClass" : "Warrior",
	"remainingCardsCount":24,
	"opponent":{
		"username":"Dummy-AI",
		"heroClass":"Hunter",
		"hp":30,
		"mp":0,
		"board":[],
		welcomeText : "Die, maggot!",
		"remainingCardsCount":24,
		"handSize" : 3
	}
}
```

Le jeu

L'objectif

Le but du jeu est de réduire la vie de l'adversaire à zéro, en attaquant avec ses cartes. Le jeu est fortement inspiré du jeu « Hearthstone » de la compagnie Blizzard.

Les cartes

L'état d'une carte sur le jeu

Une carte ayant son state « IDLE » peut être jouée, alors qu'une carte « SLEEP » ne peut être joué avant le prochain tour.

Placer une carte sur le jeu

Pour jouer une carte qui se trouve dans votre main, vous devez avoir suffisamment de « mp ».

Une carte « charge »

Ce type de carte déposée sur le jeu peut immédiatement attaquer, alors que les autres doivent attendre au tour suivant.

Une carte « taunt »

L'adversaire ne peut attaquer le héro où les autres cartes derrières les cartes « taunt ».

Temps accordé pour jouer son tour

Un joueur peut prendre jusqu'à 50 secondes pour jouer son tour. Lorsqu'il a terminé de faire ses actions, il peut cependant faire l'action « END_TURN » pour le terminer plus rapidement.

Autres notes importantes

Voici quelques notes importantes concernant le serveur Magix.

Éviter d'être banni du serveur et les délais d'appel

Pour le service « games/state » : un délai de 1 seconde minimum entre chaque appel

Il est donc nécessaire d'attendre le retour de l'appel du service avant d'en lancer un autre!

Par exemple :

```
function state() {
        $.ajax({
            url : "ajax-state.php",
            type : "POST"
        })
        .done(function (msg) {
               var reponse = JSON.parse(msg);

               // traitement ici…

               setTimeout(state, 1000); // Attendre 1 seconde avant de relancer l'appel
        })
}

setTimeout(state, 1000); // Appel initial (attendre 1 seconde)
```

Le quasi temps-réel

Étant donné qu'AJAX est utilisé au lieu des Web Sockets pour le jeu, le jeu ne fonctionne pas tout-à-fait en temps réel. C'est donc normal que lorsque l'un des joueurs fait une action, un autre joueur ne la voit pas instantanément.

Distinguer un message texte d'un objet dans une variable

Voici un exemple de code qui permet de voir si la variable est un objet ou une simple chaîne de caractères. Ça peut être utilisé pour savoir si la partie est terminé ou pas

```
if (typeof maVariable !== "object") {
        if (maVariable == "GAME_NOT_FOUND") {
               // Fin de la partie. Est-ce que j'ai gagné? Je dois appeler user-info
        }
}
else {
        // maVariable est un objet. On pourrait faire, par exemple, maVariable.game.hp ou
        // maVariable.player.mp
}
```