Magix API v1.0.3

## 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](#_Toc8638395)

[Table des matières 1](#_Toc8638396)

[Le jeu Magix 2](#_Toc8638397)

[Page d’authentification 2](#_Toc8638398)

[Page « lobby » 3](#_Toc8638399)

[Page de jeu 3](#_Toc8638400)

[L’enregistrement 4](#_Toc8638401)

[Créer son « Deck » 4](#_Toc8638402)

[Se connecter à l’API 5](#_Toc8638403)

[Les services disponibles (l’API) 6](#_Toc8638404)

[Connexion au serveur 6](#_Toc8638405)

[Déconnexion du serveur 6](#_Toc8638406)

[Créer/Joindre une partie 6](#_Toc8638407)

[Pour faire une action (jouer une carte, terminer son tour, attaquer) 7](#_Toc8638408)

[L’état de la partie en cours 8](#_Toc8638409)

[Le jeu 9](#_Toc8638410)

[L’objectif 9](#_Toc8638411)

[Les cartes 9](#_Toc8638412)

[Temps accordé pour jouer son tour 9](#_Toc8638413)

[Autres notes importantes 10](#_Toc8638414)

[Éviter d’être banni du serveur et les délais d’appel 10](#_Toc8638415)

[Le quasi temps-réel 10](#_Toc8638416)

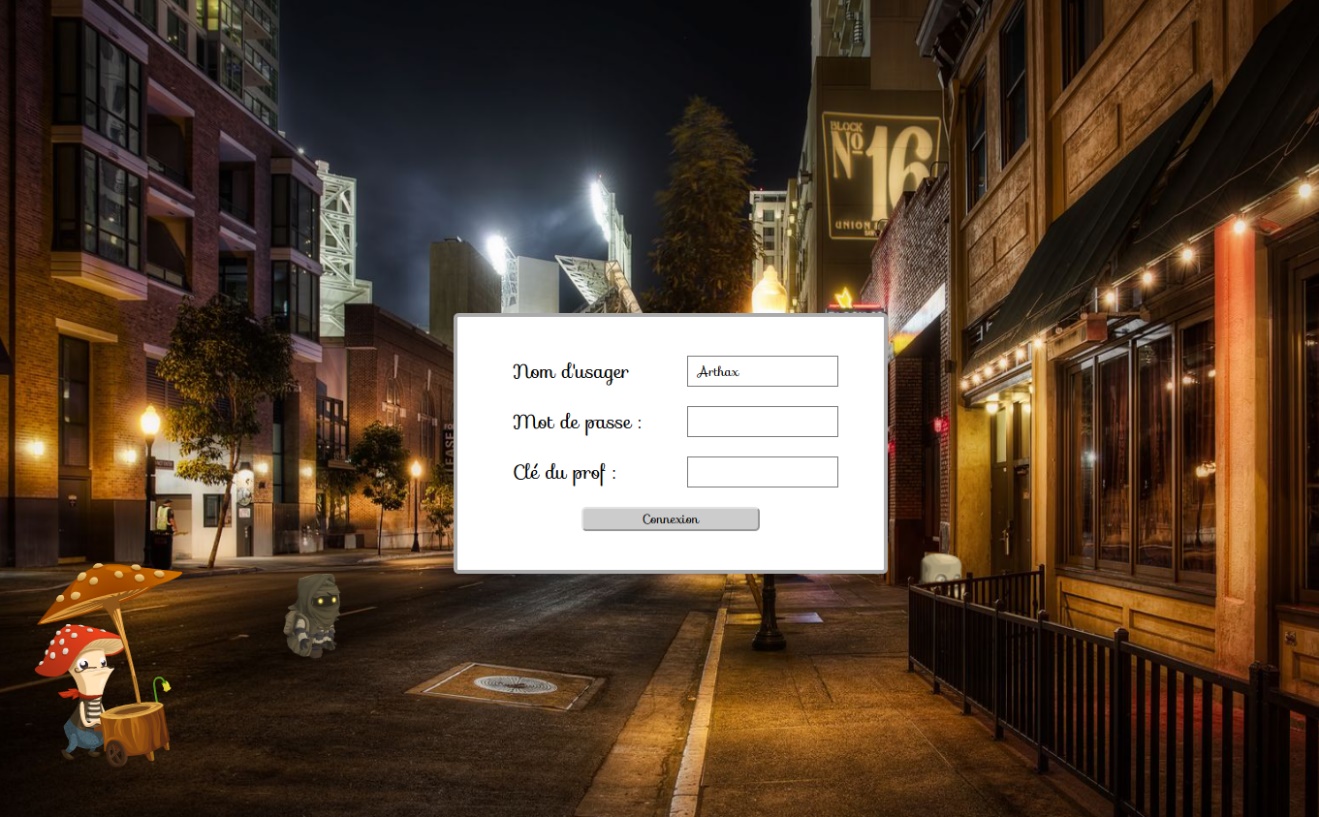
[Distinguer un message texte d’un objet dans une variable 10](#_Toc8638417)

## Le jeu Magix

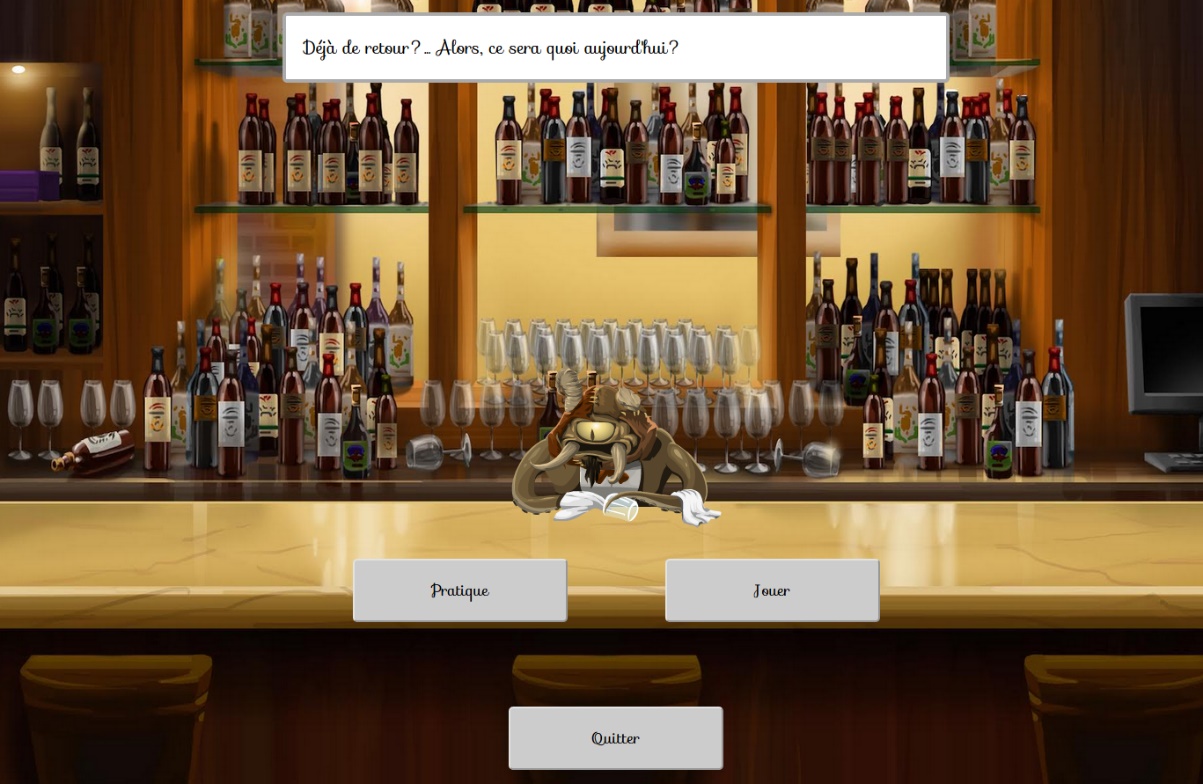
Magix est un serveur d’un jeu de cartes basé sur la technologie PHP, HTML, Node, JavaScript, MongoDB et AJAX. Celui-ci peut être joué contre l’ordinateur ou un autre joueur.

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

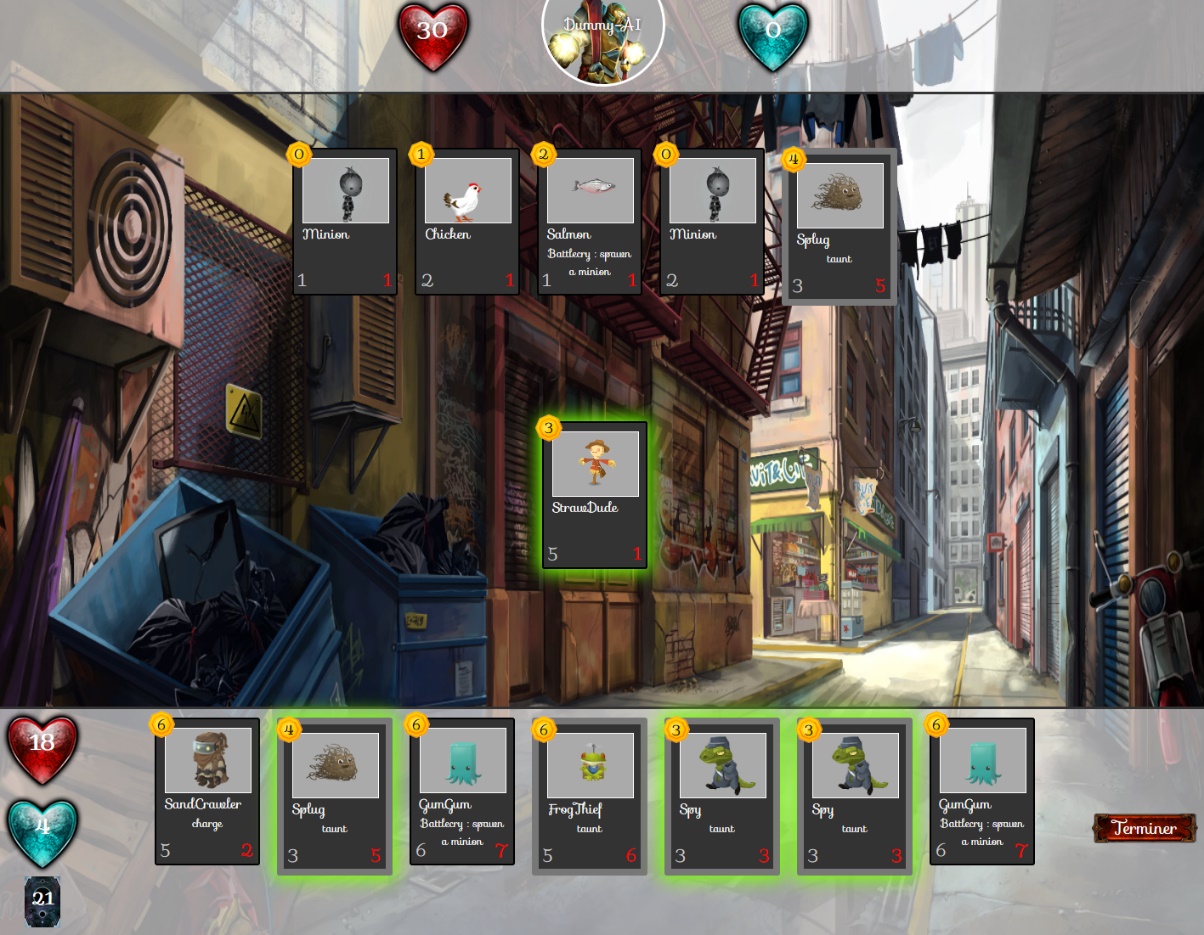
### Page d’authentification



### 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 une classe « Utils » ou 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 LoginAction ou UserDAO, …)

$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;

}

## Les services disponibles (l’API)

### Connexion au serveur

Pour faire une authentification auprès du serveur, il faut avoir complété son inscription/enregistrement (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" | - Trop de connexions rapides |

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 = array("key" => $\_SESSION["key"]);* |
| 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 | Votre clé  "PVP" ou "TRAINING" |
| Retour (succès) | "JOINED\_PVP"  "CREATED\_PVP"  "JOINED\_TRAINING" |  |
| Retour (erreur) | " INVALID\_KEY"  "INVALID\_GAME\_TYPE”  " DECK\_INCOMPLETE" |  |

Après cet appel, la partie commence, *game on!*

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

|  |  |  |
| --- | --- | --- |
| Nom du service | games/action |  |
| Paramètres | key  type  ou  type  ou  type  uid  ou  type  uid  targetuid | Votre clé  "END\_TURN"  "HERO\_POWER"  "PLAY" et  ex : 23 (identifiant unique de la carte lors de la partie)  "ATTACK"  uid de la carte  uid de la carte attaquée (ou null pour le héros adverse) |
| Retour (succès) | L’état de la partie, voir page suivante |  |
| Retour (erreur) | "INVALID\_KEY"  "ACTION\_IS\_NOT\_AN\_OBJECT"  "NOT\_ENOUGH\_ENERGY"  "BOARD\_IS\_FULL "  "CARD\_NOT\_IN\_HAND"  "CARD\_IS\_SLEEPING"  "MUST\_ATTACK\_TAUNT\_FIRST"  "OPPONENT\_CARD\_NOT\_FOUND"  "CARD\_NOT\_FOUND"  "ERROR\_PROCESSING\_ACTION"  "INTERNAL\_ACTION\_ERROR"  "HERO\_POWER\_ALREADY\_USED" | - mauvaise structure de données  - la carte coûte trop cher à jouer  - Pas assez de place pour la carte  - La carte n’est pas dans votre main  - Carte ne peut être jouée ce tour-ci  - Une carte taunt empêche ce coup  La carte attaquée n’est pas présente sur le jeu  La carte cherchée (uid) n’est pas présente  Erreur interne, ne devrait pas se produire  Autre erreur interne, ne devrait pas se produire  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,

"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

}