Projet AFJD : spécifications des « API REST »

Table of Contents

A Révisions

Date	Auteur	Révision	Contenu

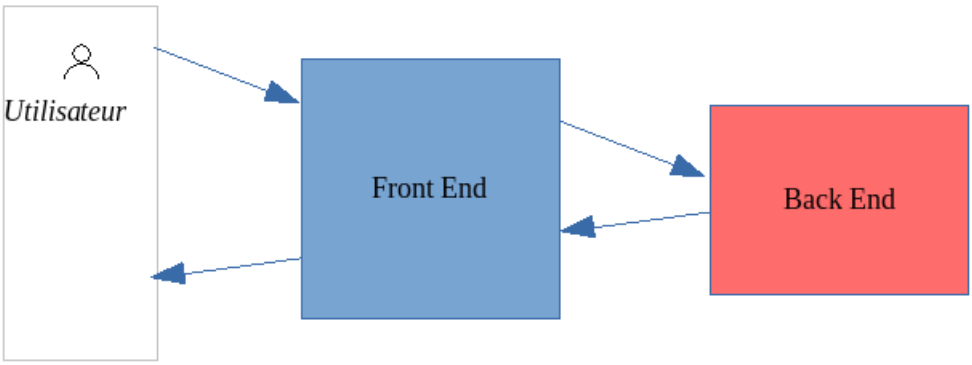
18/10/2020	JL	Draft	Première version incomplète transmise le 19/10/2020
20/102020	JL	Complète	

B Contexte

B.1 But :

Ce document présente les différents services disponibles via l'interface API REST. (« Back End »)

B.2 Vue d'ensemble simplifiée



Différence très importante :

- Un composant « Front-End » est susceptible d'être altéré à des fins malicieuses
- Un composant « Back-End » est hébergé sur un serveur contrôlé par l'association, il ne peut donc pas être altéré à des fins malicieuses.

Il faut donc toujours garder cet aspect à l'esprit lors de la conception des composants.

B.3 Les composants « Front-End »

Ce sont en réalité des Interfaces Homme Machine. Ils sont directement au contact de l'utilisateur.

Ces composants ne sont pas détaillés dans ce document.

Ces composants doivent parler à l'utilisateur en français.

B.4 Les composants « Back-End »

Ce sont, sauf exception, des serveurs REST. Ils s'appuient sur une base de données sqlite3. Les tables sqlite3 sont de deux sortes :

- Tables « objets » pour lesquelles une rangée correspond à un objet (exemple : un compte utilisateur). Chaque objet possède un identifiant numérique unique.
- Tables « relations » qui servent à relier des objets entre eux (exemple : l'appartenance à une partie). Pour ce faire, l'identifiant des objets est utilisé.

Ces composants parleront à leur client en anglais.

C Présentation des APIs

C.1 « USERS », le module identification/authentification

Ce module n’est pas REST.

Gestion des comptes utilisateurs (pseudo, mot de passe)

Important : le mot de passe n’est pas conservé en clair dans la base de données.

Ce module est une brique fondamentale de cette architecture pour éviter les usurpations d’identité.

Capacités :

- Il sait créer et conserver un couple pseudo/mot de passe (le mot de passe n'est pas conservé en clair)
- Il sait fournir un jeton lorsqu’un composant lui présente un pseudo/mot de passe qu’il reconnaît
- Il sait authentifier un jeton qui lui est présenté.
- Il logue soigneusement dans un fichier les événements qui se produisent.

Implémentation : S’appuie sur flask, flask-jwt-extended, werkzeug.security.

C.2 « EMAILS », le gestionnaire d’e-mails

Gestion des codes de confirmation des adresses mail.

Capacités :

- Il sait enregistrer un couple adresse émail et code de quatre chiffres
- Il sait authentifier un couple adresse émail et code de quatre chiffres qui lui est présenté.

Implémentation : s’appuie sur flask, flask-restful, flask_et restful.reqparse

C.3 « PLAYERS », le gestionnaire de comptes des joueurs

Gestion des joueurs avec toutes les informations à leur sujet, notamment leur fuseau horaire, leur localisation et leur adresse e-mail

Capacités :

- Il sait créer un compte utilisateur.
- Il sait modifier le compte
- Il sait supprimer le compte
- Il vérifie l'adresse de l’utilisateur par le biais d'un code de 4 chiffres que l’utilisateur doit saisir pour prouver qu’il a bien reçu le mail.
- Il peut fournir la liste des comptes

Implémentation : s’appuie sur flask, flask-restful, flask_et restful.reqparse

C.4 « SOLVER », le solveur Diplomatie

Ce composant est le plus difficile à réaliser dans l’absolu. C’est ce que l’on appelle habituellement le « moteur de résolution ». Il sait gérer toutes les variantes du jeu Diplomatie qui ne s’écartent pas

trop du modèle original, pourvu qu'on lui fournisse toutes les informations en entrée. Il récupère lui-même les informations liées à la variante.

Ce module est le cœur du système.

Capacités :

- Il sait réaliser une résolution au sens du jeu Diplomatie

Implémentation : s'appuie sur flask, flask-restful, et un solveur réalisé il y a une vingtaine d'années en langage C (éprouvé sur « stabbeurfou »)

C.5 « GAMES », le gestionnaire de parties

Ce composant est le celui qui offre le plus de services et qui sera le plus utilisé directement pour réaliser la partie de diplomatie

Capacités :

- Il sait créer une partie,
- Il sait apparier des joueurs dans la partie (et les désapparier)
- Il sait la démarrer
- Il sait y valider et déposer des ordres
- Il sait la faire avancer (réaliser une résolution à partir des ordres validé et déposés)
- Il sait y envoyer, consulter des messages diplomatiques
- Il sait y publier, consulter des presses
- Il gère les dates de visites pour déterminer les nouvelles presses et les nouveaux messages diplomatiques
- Il sait l'arrêter
- Il peut fournir la liste des parties

Implémentation : s'appuie sur flask, flask-restful, flask_restful.reqparse et flask-mail

D Interface APIs

Les interfaces grisées ne sont pas destinées à être utilisées directement (depuis le module « Front End »). Elles sont mentionnées uniquement à titre d'information.

Les cas d'erreurs grisés ne doivent pas se produire et correspondent à une sorte d'erreur interne.

La mention « protection par jeton » signifie qu'il faut présenter un jeton d'authentification pour que la requête soit acceptée. Le texte explique quel jeton est nécessaire.

La plupart des API renvoient :

- Un simple dictionnaire : {'msg' : <contenu du message en chaîne de caractère>}
- Un code HTTP

Lorsque ce n'est pas le cas, la donnée est rougie et des explications complémentaires sont fournies dans le prochain chapitre.

Sémantique des codes HTTP :

200	Opération effectuée
201	Opération effectuée, ressource ajoutée
400	Opération non réalisée
401	Opération non réalisée, absence d'identification
403	Opération non réalisée, mauvaise identification
404	Opération non réalisée, ressource absente
405	Opération non réalisée, interdite
500	Erreur interne (INUTILISÉ À AJOUTER)

D.1

D.2 « USERS » (interface standard non REST)

Pour des raisons de sécurité, la réponse à « login » reste vague et ne précise pas le défaut (utilisateur inexistant ou mauvais mot de passe)

Nom	**Méthode**	**Paramètres / Protection par jeton**	**Explications**
add	POST	user_name (str) password (str) Non	Insère un compte utilisateur dans la base. Cas d'erreur : • 'Missing JSON in request', 400 • 'Missing user_name parameter', 400 • 'Missing password parameter', 400 • 'User already exists', 400 Succès : • 'User was added', 201
remove	POST	user_name (str) Oui (il faut le jeton du compte que l'on s'apprête à supprimer)	Supprime un compte utilisateur dans la base. Cas d'erreur : • 'Missing JSON in request', 400 • 'Missing user_name parameter', 400 • 'User does not exist', 404 • 'This is not you ! Good try !', 405 Succès : • 'User was removed', 200
change	POST	user_name (str) password (str) Oui (il faut le jeton du compte que l'on s'apprête à modifier)	Modifie un compte utilisateur dans la base (le mot de passe) Cas d'erreur : • 'Missing JSON in request', 400 • 'Missing user_name parameter', 400 • 'Missing password parameter', 400 • 'User does not exist', 404 • 'This is not you ! Good try !', 405 Succès : • 'User was changed', 201

login	POST	user_name (str) password (str) Non	A partir d'un pseudo et d'un mot de passe, fournit un jeton d'authentification à utiliser par la suite Cas d'erreur : • 'Missing JSON in request', 400 • 'Missing user_name parameter', 400 • 'Missing password parameter', 400 • 'Bad user_name or password', 401 Succès : • <Le jeton d'authentification>, 200 Le jeton d'authentification est un dictionnaire : {'access_token' : <access_token>}
verify	GET	- OUI (il faut le jeton du compte dont on cherche à se prévaloir)	A partir d'un jeton d'authentification vérifie l'authentification Cas d'erreur : • Sans objet Succès : • <pseudo utilisateur>, 200 Le pseudo utilisateur est un dictionnaire : {'logged_in_as' : <pseudo>}

D.3 «EMAILS» (interface REST)

Point d'accès	**Méthode**	**Paramètres / Protection par jeton**	**Explications**
/emails	POST	email_value (str) code (int) Non	Insère un compte utilisateur dans la base. Cas d'erreur : • Sans objet Succès : • 'Email was added or updated', 201
	GET	email_value (str) code (int) Non	A partir d'un jeton d'authentification vérifie l'authentification Cas d'erreur : • 'Email {email_value} does not exists', 404 • 'Code is incorrect', 401 Succès : • 'Email is correct', 200

D.4 «PLAYERS» (interface REST)

Point d'accès	Méthode	Paramètres / Protection par jeton	Explications
/player_identifiers/ <pseudo>	GET	- Non	Renvoie le numéro (l'identifiant numérique) d'un joueur à partir de son pseudo Cas d'erreur : • Sans objet Succès : • < identifiant numérique de joueur >, 200 L'identifiant numérique de joueur est un entier.
/players/<pseudo>	GET	- Oui (le jeton du compte user sous-jacent)	Renvoie les informations relatives à un joueur à partir de son pseudo. Cas d'erreur : • 'Not allowed for retrieve!:{message}', 400 • 'Player {pseudo} doesn't exist', 404 Succès : • < informations relatives au joueur >, 200
	PUT	<informations relatives au joueur> Oui (le jeton du compte user sous-jacent)	Met à jour les informations relatives à un joueur à partir de son pseudo Cas d'erreur : • 'Not allowed for retrieve!:{message}', 400 • 'User modification failed!:{message}', 400 (mise à jour de mot de passe) • 'Not allowed for update!:{message} ', 400 • 'Player {pseudo} does not exist', 404 • 'Failed to store email code!: {message}', 400 (mise à jour de l'adresse mail) Succès : • 'Ok updated', 200 (mise à jour de mot de passe) • 'Ok but no change !', 200 (mise à jour de mot de passe) • 'Ok updated', 200
	DELETE	- Oui (le jeton du compte user sous-jacent)	Suppression d'un joueur à partir de son pseudo Cas d'erreur : • 'Player {pseudo} does not exist', 404 • 'Allocation check failed!:{message}', 400 • 'Player is still in, a game', 400 • 'User removal failed!:{message}', 400

			Succès : • ‘Ok removed‘, 200
/players	GET	- Non	Renvoie un dictionnaire identifiant numérique : pseudos de tous les joueurs Succès : • <liste des joueurs>, 200 La <liste des joueurs > est une liste de dictionnaires : <identifiant numérique> : <pseudo>}
	POST	informations relatives au joueu Non	Insère un joueur avec son pseudo et ses informations Utilise l’adresse e-mail pour envoyer un e-mail avec un code de 4 chiffres Cas d’erreur : • ‘Player {pseudo} already exists‘, 400 • ‘User creation failed!:{message}‘, 400 (mise à jour de mot de passe) • ‘Failed to store email code!: {message}‘, 400 Succès : • ‘Ok player created‘, 200
/emails	POST	Pseudo (str) email (str) Non	Vérifie un couple pseudo/code de quatre chiffres Cas d’erreur : • ‘Player {pseudo} does not exist‘, 404 • ‘Wrong code!:{message}‘, 400 Succès : • ‘Ok code is correct‘, 200

D.5 «SOLVER» (interface REST)

Point d’accès	**Méthode**	**Paramètres / Protection par jeton**	**Explications**
/solve	POST	Variant (str) advancement (int) situation (str - json) orders (str - json) role (int) names (str - json)	Réalise une résolution. « Variant » est le nom de la variante. Utiliser « standard » « Advancement » est l’avancement de la partie, il vaut zéro à la création de la partie et augmente de un à chaque résolution. Il permet de déterminer la saison (donc la phase de jeu parmi Mouvements, Retraites, Retraites + mise à jour des possessions de centres, Ajustements) et l’année (décorative) à l’aide de la variante

		Non	« Situation » est un dictionnaire : • ‘ownerships’ : <ownerships> • ‘dislodged_ones’ : <dislodged> • ‘units’ : <units> • ‘fake_units’ : <fake_units> • ‘forbiddens’ : <forbiddens> « Role » est le rôle. Une valeur différente de zéro signifie qu’il faut « inventer » des ordres pour tous les autres pays – cas de soumission d’ordres d’un joueur de la partie.

D.6 «GAMES» (interface REST)

Point d’accès	**Méthode**	**Paramètres / Protection par jeton**	**Explications**
variants/<name>	GET	- Non	Renvoie les informations de la variante Cas d’erreur : • ‘Variant {name} is incorrect as a name‘, 400 • ‘Variant {name} does not exist‘, 404 Succès : • ‘Ok code is correct‘, 200
game-identifiers/ <name>	GET	- Non	Renvoie le numéro (l’identifiant numérique) d’une partie à partir de son nom Cas d’erreur : • Sans objet Succès : • < identifiant numérique de partie >, 200 L’ identifiant numérique de partie est un entier.
/games/<name>	GET	- Non	Renvoie les informations relatives à une partie à partir de son nom Cas d’erreur : • ‘Game {name} doesn't exist‘, 404 Succès : • < informations relatives à la partie>, 200
	PUT	< informations relatives à la partie> pseudo (int)	Met à jour les informations relatives à une partie à partir de son nom Cf. POST pour les explications sur les

		Oui (le jeton du compte user sous-jacent de l'arbitre de la partie)	différents champs Cas d'erreur : • 'Game {name} does not exist', 404 • 'Need a pseudo to modify game', 400 • 'Bad authentication!:{message}', 401 • 'Failed to get id from pseudo {message', 404 • 'You do not seem to be the game master of the game', 403 • 'Not enough players !', 400 (démarrage de la partie) Succès : • 'Ok but no change !', 200 • 'Ok updated', 200
	DELETE	- Oui (le jeton du compte user sous-jacent de l'arbitre de la partie)	Suppression d'une partie à partir de son nom Cas d'erreur : • 'Game {name} does not exist ', 404 • 'Need a pseudo to delete game', 401 • 'Bad authentication!:{message} ', 401 • 'Failed to get id from pseudo {message}', 404 • 'You do not seem to be the game master of the game', 403 • Succès : • 'Ok removed', 200
games	GET	- Non	Renvoie un dictionnaire identifiant numérique : nom de toutes les parties Succès : • <liste des parties >, 200 La liste des parties est une liste de dictionnaires : { <identifiant numérique> : <nom de la partie>}
	POST	< informations relatives à la partie> pseudo (int) Oui (le jeton du compte user sous-jacent de l'arbitre de la partie)	Insère une partie avec son nom et ses informations Cas d'erreur : • 'Game {name} already exists ', 400 • 'Need a pseudo to create game', 401 • 'Bad authentication!:{message}', 401

			• ‘Failed to get id from pseudo {message}’, 404 Succès : • ‘Ok game created‘, 200
allocations	POST	game_id (int) player_id (int) pseudo (str) Oui (le jeton du compte user sous-jacent de l’arbitre de la partie ou le joueur à insérer)	Insère une allocation (une relation entre un joueur et une partie) Il faut être : • soit l’arbitre de la partie • soit le joueur concerné Cas d’erreur : • ‘Need a pseudo to join/put in game’, 401 • ‘Bad authentication!:{message}’, 401 • ‘Failed to get id from pseudo {message}’, 404 • ‘You do not seem to be either the game master of the game or the concerned player’, 403 • ‘This game is not in the proper state - please proceed to replacement (not implemented yet)’, 405 Succès : • ‘Ok allocation updated or created‘, 200
	DELETE	game_id (int) player_id (int) role_id (int) pseudo (str) Oui (le jeton du compte user sous-jacent de l’arbitre de la partie ou le joueur à retirer)	Supprime une allocation (une relation entre un joueur et une partie) Il faut être : • soit l’arbitre de la partie • soit le joueur concerné Cas d’erreur : • ‘Need a pseudo to quit/remove from game’, 401 • ‘Bad authentication!:{message}’, 401 • ‘Failed to get id from pseudo {message}’, 404 • ‘You do not seem to be either the game master of the game or the concerned player’, 403 • ‘This game is not in the proper state - please proceed to replacement (not implemented yet)’, 405 Succès : • ‘Ok allocation deleted if present‘, 200
/game-allocations/	GET	-	Renvoie la liste des allocations de la partie

<game_id>		Non	Succès : • Dictionnaire {<identifiant de joueur> : <identifiant de rôle dans la partie>}, 200
/player-allocations/ <player_id>	GET	- Non	Renvoie la liste des allocations du joueur Succès : • Dictionnaire {<identifiant de partie>: <identifiant de rôle dans la partie>}, 200
/game-positions/ <game_id>	POST	pseudo (str) center_ownerships (str - json) units (str - json) forbiddens (str _ json) Oui (le jeton du compte user sous-jacent de l'arbitre de la partie)	Altère la position de la partie (les possessions de centres, les unités et les zones interdites en retraite) Cas d'erreur : • 'Need a pseudo to rectify position in game', 401 • 'Bad authentication!:{message}', 401 • 'Failed to get id from pseudo {message}', 404 • 'You do not seem to be the game master of the game', 403 Succès : • 'Ok position rectified', 200
	GET	- Non	Renvoie la position de la partie. Cf. POST pour la description des champs. Succès : • {'ownerships : <ownerships>, 'dislodged_ones' : <dislodged_units>, 'units' : <units>, 'forbiddens' : <forbiddens>}, 200
/game-reports/ <game_id>	GET	- Non	Renvoie le dernier rapport de résolution de la partie. Succès : • {'content' : <rapport>}, 200
/game-orders/ <game_id>	POST	role_id (int) orders (str _ json) names (str - json) pseudo (str) Oui (le jeton du compte user sous-jacent du joueur associé au rôle)	Soumission d'ordres diplomatie par un joueur d'une partie. Cas d'erreur : • 'Game master cannot submit orders in game - please usurp game player (not implemented)', 400 • 'Need a pseudo to submit orders in game', 401 • 'Bad authentication!:{message}', 401

			• ‘Failed to get id from pseudo {message}’, 404 • ‘You do not seem to be the player who is in charge’, 403 • ‘Variant {variant_name} doesn't exist’, 404 • ‘Failed to submit orders {message} : {submission_report}’, 400 Succès : • ‘Ok orders submitted {submission_report}’, 201
	GET	role_id (int) pseudo (str) Oui (le jeton du compte user sous-jacent du joueur associé au rôle)	Renvoie les ordres soumis par le joueur de la partie Cas d’erreur : • ‘Need a pseudo to retrieve orders in game’, 401 • ‘Bad authentication!:{message}’, 401 • ‘Failed to get id from pseudo {message}’, 404 • ‘You do not seem to be the game master of the game’, 403 (role_id = 0) • ‘You do not seem to be the player who is in charge’, 403 (role_id != 0) • ‘Variant {variant_name} doesn't exist’, 404 • ‘Failed to submit orders {message} : {submission_report}’, 400 Succès : • {‘orders’ : <orders>, ‘fake_units’ : <unités factices>}, 200
/game-adjudications/<game_id>	POST	names (str) pseudo (str) Oui (le jeton du compte user sous-jacent de l’arbitre de la partie)	Résolution de la partie Cas d’erreur : • ‘Need a pseudo to adjudicate in game’, 401 • ‘Bad authentication!:{message}’, 401 • ‘Failed to get id from pseudo {message}’, 404 • ‘You do not seem to be the game master of the game’, 403 • ‘Variant {variant_name} doesn't exist,’ 404 • ‘Failed to adjudicate{message} : {submission_report}’, 400 Succès : • ’Ok adjudication performed and

			game updated : {adjudication_report}', 201
/simulation	POST	variant_name (str) orders (str - json) center_ownerships (str - json) units (str - json) names(str - json) Non	Simulation de résolution (« bac à sable ») Cas d'erreur : • Variant {variant_name} doesn't exist, 404 • Failed to adjudicate{message} : {submission_report}, 400 Succès : • 'Ok adjudication performed {adjudication_report}', 201
/game-messages/ <game_id>	POST	role_id (int) dest_role_id (int) content (str) pseudo (str) Oui (le jeton du compte user sous-jacent de l'arbitre ou du joueur de la partie)	Insertion d'un message diplomatique Cas d'erreur : • 'Need a pseudo to insert message in game', 401 • 'Bad authentication!:{message}', 401 • 'Failed to get id from pseudo {message}', 404 • 'You do not seem to be the game master of the game', 403 (role_id=0) • 'You do not seem to be the player who is in charge', 403 (role_id != 0) Succès : • 'Ok message inserted {content}', 201
	GET	limit (int) role_id (int) pseudo (str) Oui (le jeton du compte user sous-jacent de l'arbitre ou du joueur de la partie)	Récupération de messages diplomatiques (au plus 'limit' messages si ce paramètre est fourni) Cas d'erreur : • 'Need a pseudo to get message in game', 401 • 'Bad authentication!:{message}', 401 • 'Failed to get id from pseudo {message}', 404 • 'You do not seem to be the game master of the game', 403 (role_id=0) • 'You do not seem to be the player who is in charge', 403 (role_id != 0) Succès : • <liste de messages>, 200 La liste de messages est une liste de : • time_stamp (cf. ci-dessous)

			• numéro de l'auteur • numéro du destinataire • contenu
/game-declarations/ <game_id>	POST	role_id (int) content (str) pseudo (str) Oui (le jeton du compte user sous-jacent de l'arbitre ou du joueur de la partie)	Insertion d'une déclaration de presse Cas d'erreur : • 'Need a pseudo to insert declaration in game', 401 • 'Bad authentication!:{message}', 401 • 'Failed to get id from pseudo {message}', 404 • 'You do not seem to be the game master of the game', 403 (role_id=0) • 'You do not seem to be the player who is in charge', 403 (role_id != 0) Succès : • 'Ok declaration inserted {content}', 201
	GET	limit (int) role_id (int) pseudo (str) Oui (le jeton du compte user sous-jacent de l'arbitre ou du joueur de la partie)	Récupération de déclarations de presse (au plus 'limit' déclarations si ce paramètre est fourni) Cas d'erreur : • 'Need a pseudo to get declarations in game', 401 • 'Bad authentication!:{message}', 401 • 'Failed to get id from pseudo {message}', 404 • 'You do not seem to be the game master of the game', 403 (role_id=0) • 'You do not seem to be the player who is in charge', 403 (role_id != 0) Succès : • <liste de déclarations>, 200 La liste de déclarations est une liste de : • time_stamp (cf. ci-dessous) • numéro de l'auteur • contenu
/game-visits/ <game_id>	POST	role_id (int) pseudo (str) Oui (le jeton du compte user sous-jacent de l'arbitre ou du joueur de la partie)	Insertion d'une visite (date de visite) Cas d'erreur : • 'Need a pseudo to insert visit in game', 401 • 'Bad authentication!:{message}', 401 • 'Failed to get id from pseudo

			{message}', 404 • 'You do not seem to be the game master of the game', 403 (role_id=0) • 'You do not seem to be the player who is in charge', 403 (role_id != 0) Succès : • 'Ok visit inserted', 201
	GET	role_id (int) pseudo (str) Oui (le jeton du compte user sous-jacent de l'arbitre ou du joueur de la partie)	Récupération de visite (date de visite) Cas d'erreur : • 'Need a pseudo to retrieve last visit in game', 401 • 'Bad authentication!:{message}', 401 • 'Failed to get id from pseudo {message}', 404 • 'You do not seem to be the game master of the game', 403 (role_id=0) • 'You do not seem to be the player who is in charge', 403 (role_id != 0) Succès : • <time_stamp>, 200 Un time_stamp est un entier. Il indique la date de la dernière visite. Il est exprimé sous la forme du nombre de secondes depuis le 1/1/1970

E Détails des données élaborées

Certaines données plus volumineuses ou plus complexes nécessitent une description détaillée.

E.1 Données simples

Donnée	Explications
< informations relatives au joueur >	**pseudo** : str (limité à 12) – le pseudo du joueur **email** : str (limité à 30) – l'adresse e-mail du joueur (obligatoire) **email_confirmed** : int (ignorée au PUT, réalisée au GET) – 1 si cette adresse e-mail a été confirmée **téléphone** : str (limité à 15) – non vérifiée – le téléphone du joueur (facultatif) **family_name** : str (limité à 30) – le nom de famille du joueur (facultatif) **first_name** : str (limité à 20) – le prénom du joueur (facultatif) **country** : str (limité à 5) le code ISO du pays du joueur (obligatoire) **time_zone** : str (limité à 10) – forme « UTC _ 2 » du fuseau horaire du joueur (obligatoire) Attention : • Le champ **email_confirmed** ne peut pas être écrit, il est géré directement par le serveur • Le champ **password** est géré indépendamment, ne pas réaliser de PUT avec à la fois le mot de passe et une autre modification (seul le mot de passe sera pris en compte)
< informations relatives à la partie>	**name**: str (limité à 20) – le nom de la partie **description** : str (illimitée) **variant** str (limité à 20) TODO choix sur liste **archive** : int (bool) – cette partie est une partie d'archive, elle a été jouée en face à face et sert à juste être montrée ici **anonymous**: int (bool) – il est impossible de savoir les identités des joueurs **silent**: int (bool) – il est impossible aux joueurs de communiquer entre eux **cumulate**: int (bool) – il est autorisé de jouer plusieurs rôles **fast**: int (bool) – la résolution peut avoir lieu à n'importe quel moment hors de tout calcul de date limite **deadline**: str format date <année>-<mois>-<jour> (inutile au POST) **speed_moves**: int – nombre de jour à attendre avant une résolution de mouvements pour placer la date limite **cd_possible_moves**: int (bool) – un désordre civil est possible sur des mouvements **speed_retreats**: int – nombre de jour à attendre avant une résolution de retraites pour placer la date limite **cd_possible_retreats** : int (bool) – un désordre civil est possible sur des retraites **speed_adjustments** : int – nombre de jour date limite après résolution après des retraites **cd_possible_builds** : int (bool) – un désordre civil est possible sur des constructions **cd_possible_removals** : int (bool) – un désordre civil est il possible sur des suppressions **play_weekend**: int (bool) – les date limite peuvent se produire en fin de semaine **manual**: int (bool) – partie de tournoi, les affections de puissances aux joueurs au démarrage de la partie sont manuelles par l'arbitre **access_code** : int – code d'accès à la partie

	access_restriction_reliability : int – condition de fiabilité pour l'accès à la partie **access_restriction_regularity** : int – condition de régularité pour l'accès à la partie **access_restriction_performance** : int – condition de performance (niveau) pour l'accès à la partie **current_advancement** : int – démarre à zéro et augmente de 1 à chaque résolution (ignorée au PUT, réalisée au GET) **nb_max_cycles_to_play** : int – durée maximum de la partie **victory_centers** : int – condition de victoire absolue en nombre de centres **current_state** : int – état de la partie (0=en attente, 1 = démarrée, 2 = arrêtée) Attention • le champ **current_advancement** : ne peut pas être écrit, il est géré directement par le serveur

E.2 Données complexes

SDU Présentation de la structure des données pour les unités les ordres…

Donnée	Explications
center_ownerships (str – json)	SDU
units (str – json)	SDU
forbiddens (str _ json)	SDU
orders (str _ json)	SDU
names (str – json)	SDU

F Cas d'usage

F.1 Création d'un compte de joueur

Prérequis : Néant

```
curl http://localhost:5003/players -d 'pseudo=Tartempion&password=Tartempion&email=toto@labas.com&telephone=111&first_name=John&family_name=Doe&country=FRA&time_zone=UTC + 1' -X POST
```

```
{
   "pseudo": "Tartempion",
   "msg": "Ok player created"
}
```

F.2 Création d'une partie

Prérequis : un compte joueur a été créé (l'arbitre)

```
# format non REST

# récupéreation du token d'authentifcation
```

```
curl http://localhost:5001/login -H "Content-Type: application/json" -d
'{"user_name":"Tartempion", "password":"Tartempion"}' -X POST

ACCESS=<le token avec trois champs séparés par un point>

# création de la partie

curl http://localhost:5004/games  -H "access_token: $ACCESS" -d
'name=Raspoutine&description=test&variant=standard&archive=0&anonymous=0&silent=
0&cumulate=0&fast=0&speed_moves=2&speed_retreats=1&speed_adjustments=1&play_w
eekend=0&manual=0&access_code=0&access_restriction_reliability=0&access_restriction_r
egularity=0&access_restriction_performance=0&nb_max_cycles_to_play=500&pseudo=Tart
empion' -X POST
```

```
{
   "name": "Raspoutine",
   "msg": "Ok game created"
}
```

F.3 Appariement dans une partie

Prérequis : une partie a été créée et 7 comptes joueurs ont été créés

```
ACCESS=<le token avec trois champs séparés par un point>

# create a pairing

curl http://localhost:5004/allocations -H "access_token: $ACCESS" -d
'game_id=3&player_id=1&pseudo=Tartempion' -X POST
```

```
{
   "msg": "Ok allocation updated or created"
}
```

F.4 Démarrage d’une partie

Prérequis : Une partie a été appariée.

```
ACCESS=<le token avec trois champs séparés par un point>

# start a game

curl http://localhost:5004/games/Raspoutine -H "access_token: $ACCESS" -d
'current_state=1&pseudo=Tartempion' -X PUT
```

```
{
   "name": "Raspoutine",
   "msg": "Ok updated"
}
```

F.5 Entrée d’ordres dans une partie

Prérequis : Une partie a été démarrée.

```
curl xxx
```

```
{
```

(résultat)
}

F.6 Résolution dans une partie

Prérequis : Tous les ordres dans la partie ont été entrés.

```
curl xxx
```

{
(résultat)
}

F.7 Échange de messages diplomatiques

Prérequis : Une partie a été démarrée.

```
ACCESS=<le token avec trois champs séparés par un point>

# send message

curl http://localhost:5004/game-messages/3 -H "access_token: $ACCESS" -d
'role_id=6&dest_role_id=3&content=Time for attack&pseudo=one' -X POST

# read messages

curl http://localhost:5004/game-messages/3 -H "access_token: $ACCESS" -d
'role_id=3&pseudo=two' -X GET
```

{
"msg": "Ok message inserted : Time for attack"
}
{
"messages_list": [
[
1603226457,
6,
3,
"Time for attack"
]
]
}

F.8 Utilisation de presse

Prérequis : Une partie a été démarrée.

```
ACCESS=<le token avec trois champs séparés par un point>

# send declaration

curl http://localhost:5004/game-declarations/3 -H "access_token: $ACCESS" -d
'role_id=6&content=Hello world&pseudo=one' -X POST

# read declarations

curl http://localhost:5004/game-declarations/3 -H "access_token: $ACCESS" -d
```

'role_id=6&pseudo=one' -X GET


```
{
   "msg": "Ok declaration inserted : Hello world"
}
{
   "declarations_list": [
      [
         1603226254,
         6,
         "Hello world"
      ]
   ]
}
```


G Feuille de route

Les éléments verdis seront implémentés assez rapidement.

Évolution	**Importance**	**Effort**
Pouvoir imposer des ordres de désordre civil pour un pays Motif : *les joueurs sont parfois en retard pour entrer leurs ordres sur les parties, et il faut éviter qu'une intervention technique soit nécessaire dans ce cas*	Critique	Moyen
Bloquer les doublons sur les adresses e-mail *Motif : Cela peut entraîner des comportements déroutants pour l'utilisateur*	Critique	Faible
Récupération de mot de passe oublié par un joueur *Motif : les joueurs oublient leur mot de passe, et il faut éviter qu'une intervention technique soit nécessaire dans ce cas*	Critique	Important
Remplacement d'un joueur sur une partie en cours *Motif : les joueurs abandonnent les parties, et il faut éviter qu'une intervention technique soit nécessaire dans ce cas*	Critique	Moyen
Modification des ordres d'un joueur par l'arbitre de la partie (manière à déterminer) *Motif : les joueurs ont parfois du mal à entrer des ordres, il faut éviter qu'une intervention technique soit nécessaire dans ce cas*	Majeur	Moyen
Compte super administrateur pour attribuer le droit administrateur à certains joueurs *Motif : il faut un mode administrateur et il faut éviter qu'une intervention technique soit nécessaire pour l'attribuer*	Majeur	Faible
Mode administrateur pour usurper un compte *Motif : un joueur peut entrer un pseudo injurieux, avoir un comportement anti social, etc. Il faut éviter qu'une intervention technique soit nécessaire dans ce cas*	Majeur	Moyen
Résolution déclenchée automatiquement toutes les heures (à voir selon hébergeur) *Motif : les arbitres ne pourront pas éternellement déclencher toutes les résolutions*	Majeur	Faible
Modification de la partie nécessite un code fourni par un administrateur *Motif : un arbitre peut modifier la partie parce que la résolution ne lui convient pas. Dans ce cas soit c'est une interprétation personnelle de*	Moyen	Faible

l'arbitre (à décourager) soit un bug du module résolution (à corriger)		
Gestion du code de la partie pour réserver l'entrée à la partie *Motif : met en place le mécanisme de restriction d'accès aux parties*	Mineur	Faible
Gestion du coche « d'accord pour résoudre immédiatement » (ou « ces ordres sont définitifs ») *Motif : meilleure fluidité de la partie*	Mineur	Faible
Pouvoir d'abonner à recevoir un e-mail pour certains événements liés à la partie (déclaration, message diplomatique, résolution, résolution proche) *Motif : meilleure fluidité de la partie*	Mineur	Faible
Historique des résolutions sur une partie *Motif : les joueurs ont envie de se souvenir de ce qui s'est passé dans la partie, et permet des parties recopiées de grands éventements en face à face*	Mineur	Moyen
Évaluer pour un joueur selon des formules mathématiques à déterminer : • sa fiabilité • sa régularité • sa performance (son niveau) *Motif : ces valeurs seront des critères d'acceptations des joueurs dans les parties*	Mineur	Faible
Pouvoir tout de même accéder à certaines informations sur un joueur qui doivent être annoncées publiques : • zone géographique • fuseau horaire *Motif : ces informations ont vocation à être publiques*	Mineur	Faible
Entrée d'ordres par e-mail *Motif : à définir, permettrait aux joueurs qui ont la nostalgie des « judge » de participer à l'expérience…*	Mineur	Important
Module Tournoi	Critique	Important

H Cahier des charges

Ci-dessous la liste des exigences pour l'IHM de jeu. Les exigences sont numérotées de 10 en 10 pour permettre des insertions par la suite. La liste des exigences n'est pas exhaustive pour le projet mais l'est pour une première version complète permettant de jouer dans des conditions minimales.

REQ_10	Créer un compte de joueur en maîtrisant tous les paramètres du joueur
REQ_20	Modifier son compte de joueur déjà créée
REQ_30	Entrer le code de confirmation d'adresse e-mail pour un compte de joueur
REQ_40	Modifier son mot de passe pour un compte de joueur
REQ_50	Cacher et répéter une fois toute entrée de mot de passe (pour éviter les fautes de frappe non détectées)
REQ_60	Supprimer son compte de joueur
REQ_70	Créer une partie en maîtrisant tous les paramètres de la partie
REQ_80	Modifier une partie déjà créée
REQ_90	Mettre un joueur dans une partie (par l'arbitre de la partie ou le joueur)
REQ_100	Retirer un joueur d'une partie (par l'arbitre de la partie ou le joueur)

REQ_110	Démarrer une partie
REQ_120	Arrêter une partie
REQ_130	Afficher la liste des parties
REQ_140	Choisir une partie et charger les éléments de cette partie
REQ_150	Recharger les éléments de la partie en cours
REQ_160	Entrer des ordres dans une partie dans au moins un des modes suivants : • texte • choix sur liste (le plus facile) • souris (de préférence) et les valider sur le serveur
REQ_170	Poster une déclaration dans une partie
REQ_180	Envoyer un message diplomatique dans une partie
REQ_190	Lire les déclarations d'une partie
REQ_200	Lire un message diplomatique d'une partie
REQ_210	Être informé d'une nouvelle déclaration
RES_220	Être informé d'un nouveau message diplomatique
REQ_230	Modifier une position (les centres)
REQ_240	Modifier une position (les unités)
REQ_250	Réaliser une simulation à partir d'une position et des ordres et visualiser le résultat
REQ_260	Réaliser la résolution d'une partie pour toutes les phases (mouvements, retraites, ajustements) avec tous les types d'ordres et visualiser le résultat
	(À COMPLÉTER SI BESOIN)

I Fournitures

I.1 Variante standard

Le fichier « **standard.json** » est fourni à titre indicatif, mais son contenu devra s'obtenir par une requête au serveur de parties sur la variante. Il définit complètement la variante « standard »

Les explications ci-dessous aident à en comprendre le fonctionnement (valable pour toutes les variantes)

On se base sur la carte de la boîte de jeu Hasbro de référence :

1	ADR	13	BOT	25	GAL	37	mar	49	PIE	61	STP
2	AEG	14	BRE	26	GAS	38	MID	50	POR	62	swe
3	alb	15	BUD	27	GOL	39	MOS	51	PRU	63	SYR
4	ank	16	BUL	28	GRE	40	MUN	52	ROM	64	TRI
5	APU	17	bur	29	HEL	41	naf	53	ruh	65	TUN
6	ARM	18	cly	30	hol	42	NAP	54	rum	66	tus
7	BAL	19	CON	31	ION	43	NAT	55	SER	67	TYN
8	BAR	20	den	32	IRI	44	NRG	56	sev	68	TYR
9	BEL	21	EAS	33	KIE	45	NTH	57	SIL	69	UKR
10	BER	22	EDI	34	lon	46	nwy	58	SKA	70	VEN
11	BLA	23	ENG	35	lvn	47	PAR	59	SMY	71	VIE
12	BOH	24	FIN	36	lvp	48	PIC	60	spa	72	wal

73 war
74 WES
75 YOR

Par convention :

Types d'unité	
1	armée
2	flotte

Types d'unités	
1	côte
2	terre
3	mer

Toutes les numérotations commencent à 1.

Clé	Information	Description
regions	*les régions*	*Une liste d'entiers entre 1 et 3 dénotatnt le type de la région* *Important : La taille de la liste donne le nombre de régions*
centers	*les centres*	*Une liste d'entier indiquant dans la table des régions celles qui sont des centres.*
roles	*les rôles*	*Une valeur 'number' pour le nombre de rôles (7)*
start_centers	*les centres de départ*	*Une liste pour chaque rôle de liste de centres de départ (numérotés par raport à la liste des centres)*
type_coasts	*les types de côtes*	*Une valeur 'number' pour le nombre de types de côtes (3)*
coastal_zones	*les zones côtières*	*Une liste de listes à deux éléments : le numéro de région et le numéro de type de côte* *(Ce sont les côtes supplémentaires incluses dans les régions côtières à plusieurs orientations)*
start_units	*les unités de départ*	*Une liste pour chaque rôle de dictionnaires : type d'unité, liste des zones de localisation*
year_zero	*l'année zéro*	*L'année précédente à celle du premier tour de jeu*
neighbouring	*les voisinages*	*Une liste par type d'unité de dictionnaires par zone de liste de zones atteignables*
distancing	*les éloignements*	*Une liste par rôle de dictionnaire par type d'unité de dictionnaires par zone de distances*

I.2 Présentation de la carte de jeu

Les fichiers suivants sont fournis pour aider à mettre en place la présentation à l'écran de la variante « standard »

Ils correspondent à la boite de jeu Hasbro de référence.

Nom	Description
centers.ini	La position des différents centres sur la carte sous la forme suivante : **[1]** **x_pos=532** **y_pos=449** **; ank** Il y a 34 centres.

coasts.ini	Le nom des différentes côtes sous la forme suivante : **[1]** **name=ec** Il y a 6 côtes.
map.png	La carte graphique au format 630 pixels (largeur) x 535 pixels (hauteur)
orders.ini	Le nom des différents types ordres sous la forme suivante : **[1]** **name=Attaquer** **; -** Il y a 9 types d'ordres : attaquer, soutenir offensivement ,soutenir défensivement, tenir, convoyer, retraiter, disperser, construire, supprimer.
roles.ini	Les informations sur les différents rôles sous la forme suivante (nom, nom passif, lettre, couleur) : **[1]** **name=Angleterre** **adjective_name=anglais** **letter_name=E** **red=0** **green=0** **blue=255** **; dark blue for England** Il y a 7 rôles : Angleterre, France, Allemagne, Italie, Autriche, Russie et Turquie.
seasons.ini	Le nom des différentes saisons sous la forme suivante : **[1]** **name=Printemps** Il y a 5 saisons : Printemps, Été, Automne, Hiver, Bilan.
units.ini	Le nom des différents types d'unités sous la forme suivante : **[1]** **name=Armee** Il y a 2 types d'unité : Armée et Flotte.
zones.ini	Le nom des différentes zones sous la forme suivante (nom, placement du nom, placement de l'unité) : **[1]** **name=ADR** **x_name=325** **y_name=409** **x_pos=342** **y_pos=429** **;ADR** Il y a 81 zones : 75 régions et 6 côtes spéciales.

I.3 Pays des joueurs

Le fichier « **country_list.csv** » sert à sélectionner dans l'interface le pays des joueurs. Chaque ligne définit donc un pays sélectionnable.

	Champ 1	**Champ 2**
	Code du pays à trois lettres à transmettre au serveur / recevoir du serveur	*Nom du pays à afficher à l'utilisateur*
Exemple	FRA	*France*

I.4 Fuseau horaire des joueurs :

Le fichier « **timezone_list.csv** » sert à sélectionner dans l'interface le fuseau horaire des joueurs. Chaque ligne définit donc un fuseau horaire sélectionnable.

	Champ 1	**Champ 2**
	Code du fuseau horaire à transmettre au serveur / recevoir du serveur	*Description de la time zone à afficher à l'utilisateur*
Exemple	UTC + 1	*Tunis, Casablanca, Lagos, Berlin, Vienne, Rome, Oslo, PARIS*