Projet AFJD : spécifications des « API REST »

Table des matières

A. Révisions

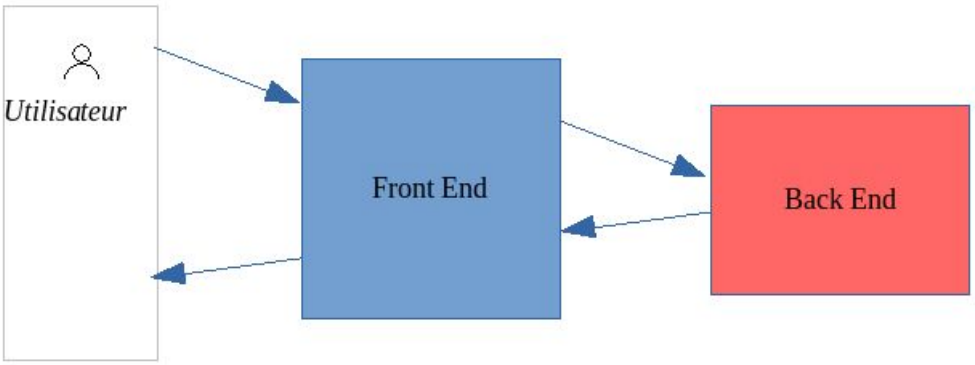
Date	Auteur	Révision	Contenu
18/11/2020	JL	Draft	Première version incomplète transmise le 22/10/2020
14/11/2020	JL	Draft	Première version complète transmise le 14/11/2020

A. Contexte

A.1. But

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

A.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.

A.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.

A.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.

B. Présentation des APIs

B.1. « USERS », le module identification/authentication

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 enregistre dans des logs soigneusement dans un fichier les événements qui se produisent.

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

B.2. « EMAILS », le gestionnaire d'e-mails

Gestion des codes de confirmation des adresses mail.

Capacités :

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

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

B.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

B.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 »)

B.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

C. 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)

C.1. « 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)	Modifie un compte utilisateur dans la base (le mot de passe)

		password (str) Oui (il faut le jeton du compte que l'on s'apprête à modifier)	 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 : {'AccessToken' : <AccessToken' >}
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>}

C.2. «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

C.3. «PLAYERS» (interface REST)

Point d'accès	**Méthode**	**Paramètres / Protection par**	**Explications**

		jeton	
/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

C.4. «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) Non	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 « Situation » est un dictionnaire : ● 'ownerships' : <ownerships> ● 'dislodged_ones' : <dislodged> ● 'units' : <units> ● 'fake_units' : <fake_units> ● 'forbiddens' : <forbiddens> « Orders » est une liste de : ● <order> « 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.

Ce module n'étant pas appelable directement depuis les IHM, ses paramètres ne seront pas détaillés.

C.5. «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) Oui (le jeton du compte user sous-jacent de l'arbitre de la partie)	Met à jour les informations relatives à une partie à partir de son nom Cf. POST pour les explications sur les 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 du 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 du 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/<game_id>	GET	- Non	Renvoie la liste des allocations de la partie 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 (délogées comprises) 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 diplomatiques 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 nombre. Il indique la date de la dernière visite. Il est exprimé sous la forme du nombre de secondes depuis le 1/1/1970

D. Détails des données élaborées

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

D.1. Données non liées au jeu (le joueur & la partie)

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 **telephone** : 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 – forcé à « standard » pour le moment **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 dates limites 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

D.2. Généralités sur les données liées au jeu (positions, ordres)

D.2.1. Un ordre

Un ordre est synthétisé de la manière suivante :

- Le rôle, c'est-à-dire le code du pays, par exemple pour la variante standard :

1	Angleterre
2	France
3	Allemagne
4	Italie
5	Autriche
6	Russie
7	Turquie

- Le type d'ordre, c'est-à-dire quel ordre il s'agit :

Phase de mouvements	1	Attaquer
	2	Soutenir offensivement
	3	Soutenir défensivement
	4	Tenir
	5	Convoyer
Phase de retraite	6	Retraiter
	7	Disperser (ne pas retraiter)
Phase d'ajustement	8	Construire
	9	Retirer

- La zone de l'unité active : l'unité active est celle qui reçoit l'ordre.
- La zone de l'unité passive : l'unité passive est celle qui est soutenue ou convoyée (elle n'existe pas forcément)
- La zone destination : la zone de destination est la destination de l'attaque ou du soutien offensif ou du convoi (elle n'existe pas forcément)

Tableau récapitulatif :

Ordre formulé de manière classique	Unité active	Unité passive	Zone destination
A PAR - BUR	PAR		BUR
A PAR S A MAR - BUR	PAR	MAR	BUR
A PAR S A BUR	PAR	BUR	
A PAR T	PAR		
F MID C A BRE - SPA	MID	BRE	SPA
A PAR R BUR	PAR		BUR
A PAR A	PAR		
+ A PAR	PAR (dans ce cas l'unité est une unité « fictive » dans une liste à part)		
- A PAR	PAR		

Une unité est caractérisée par le numéro de la zone dans laquelle elle se trouve. Se reporter au paragraphe de la variante dans le chapitre « **Fournitures** » ci-dessous.

D.2.2. Une possession

Une possession est synthétisée de la manière suivante :

- le rôle, c'est-à-dire le code du pays (cf ci-dessus)
- le numéro du centre

D.2.3. Une unité

Se présentent en réalité trois types différents d'unités :

- Une unité traditionnelle est une unité qui existe sur le plateau de jeu.
- Une unité délogée est une unité qui doit faire retraite, elle est représentée de manière différente et inclut l'information de l'origine de celle qui l'attaque (car elle ne peut faire retraite vers cette direction).
- Une unité factice est une unité qui n'existe pas encore mais est référencée dans un ordre de construction.

Une unité est synthétisée de la manière suivante :

- le rôle, c'est-à-dire le code du pays (cf ci-dessus)
- la région qu'elle occupe (ce peut être une région « spéciale)
- pour une délogée : l'origine de l'attaque (la région d'où elle vient – ce ne peut pas être une région « spéciale)

D.2.4. Une interdiction

Une interdiction est une région « bloquée » où il est interdit de faire retraite du fait d'un conflit d'accès à la phase précédente (de mouvements)

Une interdiction est synthétisée de la manière suivante :

- le numéro de la région (ce ne peut pas être une région « spéciale)

D.3. Passage des noms

Il est parfois nécessaire de passer au serveur un dictionnaire des noms (« names ») utilisés. Ces noms seront utilisés dans le compte rendu de résolution en retour fourni par le serveur. La seule contrainte sur ces noms est leur unicité : deux rôles, deux zones ou deux cotes identiques empêcheront la résolution.

La donnée nom (« names ») est un dictionnaire :

- « roles » → un dictionnaire de : numéro → tuple (nom du rôle, nom de la nationalité, code initiale)
- « zones » → un dictionnaire de numéro → nom de la zone (ou une chaîne vide pour une région spéciale)
- « coasts » → un dictionnaire de numéro → nom de la côte

Exemple de « roles »à fournir :

Numéro	Nom du rôle	Nom de la nationalité	Code initiale
0	Arbitre	arbitre	M
1	Angleterre	anglais	E
2	France	français	F
3	Allemagne	allemand	G
4	Italie	italien	I
5	Autriche	autrichien	A
6	Russie	russe	R
7	Turquie	turc	T

Le nom de nationalité servira pour rendre compte du soutien offensif ou du convoi d'une unité d'au autre rôle:

A VEN S **allemand** A MUN - TYR

Exemple de « zones » à fournir :

Se reporter au paragraphe de la variante dans le chapitre « **Fournitures** » ci-dessous.

Exemple de « coasts »à fournir :

Numéro	Nom côte
1	ec
2	nc
3	sc

D.4. Cas par cas sur les données liées au jeu (positions, ordres)

D.4.1. Altération de position

Le point d'accès concerné est :

/game-positions/<game_id>	POST	center_ownerships (str - json) units (str - json) forbiddens (str _ json)

Les possessions (« center_ownerships ») sont sous la forme d'une liste de dictionnaire : "role"→ <numéro du rôle>; "center_num"→ <numéro du centre>

Les unités (« units ») sont sous la forme d'une liste de dictionnaire : "type_unit" → <numéro du type d'unité>; "role"→<numéro du rôle>; "zone"→ <numéro de zone> et éventuellement "dislodged_origin" → <numéro de région> pour une unité délogée

Les interdits (« forbiddens ») sont sous la forme d'une liste de : <numéro de régions>

D.4.2. Récupération de position

Le point d'accès concerné est :

/game-positions/<game_id>	GET	{'ownerships : <ownerships>, 'dislodged_ones' : <dislodged_units>, 'units' : <units>, 'forbiddens' : <forbiddens>}

Les possessions (« ownerships ») sont sous la forme d'un dictionnaire : <numéro du centre> → <numéro du rôle>

Les unités délogées (« dislodged_ones ») sont sous la forme d'un dictionnaire : <numéro du rôle> → tuple (<numéro du type d'unité>, <numéro de la région de l'unité>, <numéro de région d'origine de l'unité attaquante>)

Les unités (« units ») sont sous la forme d'un dictionnaire : <numéro du rôle> → liste de tuple

(<numéro du type d'unité>, <numéro de région de l'unité>)

Les interdits (« forbiddens ») sont sous la forme d'une liste de : <numéro de région>

D.4.3. Soumission d'ordres

Le point d'accès concerné est :

/game-orders/<game_id>	POST	{'orders' : <orders>, 'fake_units' : <unités factices>}

Les ordres (« orders ») sont sous la forme d'une liste de dictionnaire : "order_type" → <numéro du type de l'ordre>; "active_unit"→ (description de l'unité); "passive_unit"→ (description de l'unité); "destination_zone" → <numéro de la région de destination>

Les unités factices (« fake_units ») sont sous la forme d'une liste de : (description de l'unité)

Description d'une unité : dictionnaire "type_unit" → <numéro du type de l'unité>; "role" → <numéro du rôle du

propriétaire>; “zone” : <numéro de la région occupée>.

D.4.4. Récupération d’ordres

Le point d’accès concerné est :

/game-orders/<game_id>	GET	{‘orders’ : <orders>, ‘fake_units’ : <unités factices>}

Les ordres (« ordres ») sont sous la forme d’une liste de : tuple (<numéro de partie> <numéro du rôle> <numéro du type de l’ordre> <numéro de la région de l’unité active> <numéro de la région de l’unité passive (ou zéro)> <numéro de la région destination (ou zéro)>)

Les unités factices (« fake_units ») sont sous la forme d’une liste de : tuple (<numéro de partie> <numéro du type de l’unité> <numéro de la région occupée> <numéro du rôle> <chiffre zéro> <chiffre un>)

D.4.5. Simulation d’ordres sur une situation

Ce cas figure mélange la rectification de position et la soumission d’ordres.

Le point d’accès concerné est :

/simulation	POST	variant_name (str) orders (str - json) center_ownerships (str - json) units (str - json)

Pour les ordres, se reporter à la soumission d’ordres ci-dessus.

Pour la situation, c'est-à-dire (« center_ownerships ») et (« units ») , se reporter à l’altération de position ci-dessus.
Puisqu’une simulation porte toujours sur des ordres de mouvements la donnée (« forbiddens ») n’est pas utilisée/

E. Cas d'usage

E.1. Création d'un compte joueur

Prérequis : Néant

```
# creation compte

curl https://afjd3.eu.ngrok.io:443/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"
}
```

E.2. Création d'une partie

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

```
# format non REST

# récupération du token d'authentification

curl https://afjd1.eu.ngrok.io:443/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 https://afjd4.eu.ngrok.io:443/games  -H "AccessToken: $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_weekend=0&manual=0&access_code=0&access_restriction_reliability=0&access_restriction_regularity=0&access_restriction_performance=0&nb_max_cycles_to_play=500&pseudo=Tartempion' -X POST
```

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

E.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>

# creation d'un appariement

curl https://afjd4.eu.ngrok.io:443/allocations -H "AccessToken: $ACCESS" -d 'game_id=3&player_id=1&pseudo=Tartempion' -X POST
```

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

E.4. Démarrage d'une partie

Prérequis : Une partie a été appariée.

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

 # demarrage partie

curl https://afjd4.eu.ngrok.io:443/games/Raspoutine -H "AccessToken: $ACCESS" -d 'current_state=1&pseudo=Tartempion' -X PUT


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


E.5. Entrée d'ordres dans une partie

Prérequis : Une partie a été démarrée.

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

 # soumission ordres

curl https://afjd4.eu.ngrok.io/game-orders:443/3 -H "AccessToken: $ACCESS" -d 'names={"roles":{"0":["Arbitre","Arbitre","M"],"1":["Angleterre","anglais","E"],"2":["France","francais","F"],"3":["Allemagne","allemand","G"],"4":["Italie","italien","I"],"5":["Autriche","autrichien","A"],"6":["Russie","russe","R"],"7":["Turquie","turc","T"]},"zones":{"1":"ADR","2":"AEG","3":"alb","4":"ank","5":"apu","6":"arm","7":"BAL","8":"BAR","9":"bel","10":"ber","11":"BLA","12":"boh","13":"BOT","14":"bre","15":"bud","16":"bul","17":"bur","18":"cly","19":"con","20":"den","21":"EAS","22":"edi","23":"ENG","24":"fin","25":"gal","26":"gas","27":"GOL","28":"gre","29":"HEL","30":"hol","31":"ION","32":"IRI","33":"kie","34":"lon","35":"lvn","36":"lvp","37":"mar","38":"MID","39":"mos","40":"mun","41":"naf","42":"nap","43":"NAT","44":"NRG","45":"NTH","46":"nwy","47":"par","48":"pic","49":"pie","50":"por","51":"pru","52":"rom","53":"ruh","54":"rum","55":"ser","56":"sev","57":"sil","58":"SKA","59":"smy","60":"spa","61":"stp","62":"swe","63":"syr","64":"tri","65":"tun","66":"tus","67":"TYN","68":"tyr","69":"ukr","70":"ven","71":"vie","72":"wal","73":"war","74":"WES","75":"yor","76":"","77":"","78":"","79":"","80":"","81":""},"coasts":{"1":"ec","2":"nc","3":"sc"}}&orders=[{"order_type":1,"active_unit":{"type_unit":2,"role":3,"zone":20},"destination_zone":62},{"order_type":1,"active_unit":{"type_unit":1,"role":3,"zone":51},"destination_zone":35},{"order_type":4,"active_unit":{"type_unit":1,"role":3,"zone":12}}]&role_id=3&pseudo=two' -X POST


```
{
   "msg": "Ok orders submitted SOLVEUR : INFORMATION : R\u00e9solution ou validation d'ordres pour Mouvements automne 1901\n\n"
}
```


E.6. Résolution dans une partie

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

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

 # resolution partie

curl https://afjd4.eu.ngrok.io/game-adjudications:443/3 -H "AccessToken: $ACCESS" -d 'names={"roles":{"0":["Arbitre","Arbitre","M"],"1":["Angleterre","anglais","E"],"2":["France","francais","F"],"3":["Allemagne","allemand","G"],"4":["Italie","italien","I"],"5":["Autriche","autrichien","A"],"6":["Russie","russe","R"],"7":["Turquie","turc","T"]},"zones":{"1":"ADR","2":"AEG","3":"alb","4":"ank","5":"apu","6":"arm","7":"BAL","8":"BAR","9":"bel","10":"ber","11":"BLA","12":"boh","13":"BOT","14":"bre","15":"bud","16":"bul","17":"bur","18":"cly","19":"con","20":"den","21":"EAS","22":"edi","23":"ENG","24":"fin","25":"gal","26":"gas","27":"GOL","28":"gre","29":"HEL","30":"hol","31":"ION","32":"IRI","33":"kie","34":"lon","35":"lvn","36":"lvp","37":"mar","38":"MID","39":"mos","40":"mun","41":"naf","42":"nap","43":"NAT","44":"NRG","45":"NTH","46":"nwy","47":"par","48":"pic","49":"pie","50":"por","51":"pru","52":"rom","53":"ruh","54":"rum","55":"ser","56":"sev","57":"sil","58":"SKA","59":"smy","60":"spa","61":"stp","62":"swe","63":"syr","64":"tri","65":"tu

```
n","66":"tus","67":"TYN","68":"tyr","69":"ukr","70":"ven","71":"vie","72":"wal","73":"war","74":"
WES","75":"yor","76":"","77":"","78":"","79":"","80":"","81":""},"coasts":{"1":"ec","2":"nc","3":"
sc"}}&pseudo=Tartempion' -X POST
```


```
{
   "msg": "Ok adjudication performed and game updated : SOLVEUR : INFORMATION : R\u00e9solution ou validation
d'ordres pour Mouvements printemps 1901\n\n"
}
```


E.7. Échange de messages diplomatiques

Prérequis : Une partie a été démarrée.

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

# envoi message

curl https://afjd4.eu.ngrok.io/game-messages:443/3 -H "AccessToken: $ACCESS" -d
'role_id=6&dest_role_id=3&content=Time for attack&pseudo=one' -X POST

# read messages

curl https://afjd4.eu.ngrok.io/game-messages:443/3 -H "AccessToken: $ACCESS" -d 'role_id=3&pseudo=two'
-X GET
```


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


E.8. Utilisation de presse

Prérequis : Une partie a été démarrée.

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

# emission declaration

curl https://afjd4.eu.ngrok.io:443/game-declarations/3 -H "AccessToken: $ACCESS" -d
'role_id=6&content=Hello world&pseudo=one' -X POST

# read declarations

curl https://afjd4.eu.ngrok.io/game-declarations/3 -H "AccessToken: $ACCESS" -d 'role_id=6&pseudo=one'
-X GET
```


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

```
      ]
    ]
  }
```

F. Feuille de route

Les éléments verdis seront implémentés assez rapidement.

	Évolution	Importance	Effort
1.	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
2.	Bloquer les doublons sur les adresses e-mail *Motif : Cela peut entraîner des comportements déroutants pour l'utilisateur*	Critique	Faible
3.	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
4.	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
5.	Enregistrer la date d'identification d'un joueur et pouvoir accéder à cette information *Motif : on veut savoir si les joueurs sont actifs ou pas*	Moyen	Moyen
6.	Enregistrer la validation des ordres par un joueur sur une partie et pouvoir accéder à cette information *Motif : l'arbitre doit savoir quels sont les ordres validés*	Moyen	Moyen
7.	Pouvoir entrer des ordres de communication *Motif : les joueurs veulent dans des parties silencieuses entrer un ordres a par s a lon pour communiquer d'une certaine manière*	Moyen	Moyen
8.	Pouvoir visualiser la résolution précédente (donc avoir accès à la situation et les ordres) *Motif : les joueurs veulent la visualiser graphiquement*	Moyen	Moyen
9.	Envoyer un email sur résolution *Motif : les joueurs veulent être avertis rapidement de l'évolution de la partie*	Moyen	Moyen
10.	Prévoir les votes de joueurs. Typiquement (mais pas seulement) l'arrêt de la partie	Moyen	Faible
11.	Afficher les choix de résolution par rapport au DATC par requête REST (à prévoir dans le front end) *Motif : Cette information quoique très technique peut s'avérer interessante*	Moyen	Moyen
12.	Dissocier le champ "pays" de la donnée joueur en deux : la résidence et la nationalité (avec les mêmes conventions de valeurs)	Faible	Faible
13.	Empêcher d'appeler directement le solveur *Motif : évite une application parasite qui l'utilise indépendamment pour réaliser les résolutions*	Mineur	Faible
14.	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
15.	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
16.	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*	Majeur	Moyen

	cas		
17.	Résolution déclenchée automatiquement toutes les heures (à voir selon hébergeur) *<u>Motif :</u> les arbitres ne pourront pas éternellement déclencher toutes les résolutions*	Majeur	Faible
18.	Modification de la partie nécessite un code fourni par un administrateur *<u>Motif :</u> un arbitre peut modifier la partie parce que la résolution ne lui convient pas. Dans ce cas soit c'est une interprétation personnelle de l'arbitre (à décourager) soit un bug du module résolution (à corriger)*	Moyen	Faible
19.	Gestion du code de la partie pour réserver l'entrée à la partie *<u>Motif :</u> met en place le mécanisme de restriction d'accès aux parties*	Mineur	Faible
20.	Pouvoir s'abonner à recevoir un e-mail pour certains événements liés à la partie (déclaration, message diplomatique, résolution, résolution proche) *<u>Motif :</u> meilleure fluidité de la partie*	Mineur	Faible
21.	Historique des résolutions sur une partie *<u>Motif :</u> les joueurs ont envie de se souvenir de ce qui s'est passé dans la partie, et permet des parties copiées de grands événements en face à face*	Mineur	Moyen
22.	Évaluer pour un joueur selon des formules mathématiques à déterminer : ● sa fiabilité (nombre de retards) ● sa régularité (nombre de parties) ● sa performance (son niveau) *<u>Motif :</u> ces valeurs seront des critères d'acceptations des joueurs dans les parties*	Mineur	Faible
23.	Entrée d'ordres par e-mail *<u>Motif :</u> à définir, permettrait aux joueurs qui ont la nostalgie des « judge » de participer à l'expérience…*	Mineur	Important
	Module Tournoi	Critique	Important

G. 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 par le joueur)
REQ_100	Retirer un joueur d'une partie (par l'arbitre de la partie ou par 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)

H. Fournitures

H.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 et on utilise les codes suivants pour les régions :

1	ADR	21	EAS	41	naf	61	STP
2	AEG	22	EDI	42	NAP	62	swe
3	alb	23	ENG	43	NAT	63	SYR
4	ank	24	FIN	44	NRG	64	TRI
5	APU	25	GAL	45	NTH	65	TUN
6	ARM	26	GAS	46	nwy	66	tus
7	BAL	27	GOL	47	PAR	67	TYN
8	BAR	28	GRE	48	PIC	68	TYR
9	BEL	29	HEL	49	PIE	69	UKR
10	BER	30	hol	50	POR	70	VEN
11	BLA	31	ION	51	PRU	71	VIE
12	BOH	32	IRI	52	ROM	72	wal
13	BOT	33	KIE	53	ruh	73	war
14	BRE	34	lon	54	rum	74	WES
15	BUD	35	lvn	55	SER	75	YOR
16	BUL	36	lvp	56	sev		
17	bur	37	mar	57	SIL		
18	cly	38	MID	58	SKA		
19	CON	39	MOS	59	SMY		
20	den	40	MUN	60	spa		

On ajoute 6 régions spéciales :

76	BUL ec	78	SPA nc	81	STP nc
77	BUL sc	79	SPA sc	82	STP sc

Par convention pour les types on utilise les codes suivants :

Types d'unité	
1	armée
2	flotte

Types de région

1	côte
2	terre
3	mer

La variante est définie par :

Clé	Information	Description
regions	*les régions*	*Une liste d'entiers entre 1 et 3 dénotant 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 rapport 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*

La table des centres :

1	ank	10	gre	19	nwy	28	stp
2	bel	11	hol	20	par	29	swe
3	ber	12	kie	21	por	30	tri
4	bre	13	lon	22	rom	31	tun
5	bud	14	lvp	23	rum	32	ven
6	bul	15	mar	24	ser	33	vie
7	con	16	mos	25	sev	34	war
8	den	17	mun	26	smy		
9	edi	18	nap	27	spa		

H.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 boîte 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, faire retraite, 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.

H.3. Pays de résidence 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*

H.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 du fuseau horaire à afficher à l'utilisateur*
Exemple	UTC + 1	*Tunis, Casablanca, Lagos, Berlin, Vienne, Rome, Oslo, PARIS*

-