PolyDiff

Protocole de communication

Version 1.0

Historique des révisions

| **Date** | **Version** | **Description** | **Auteur** |
| --- | --- | --- | --- |
| 2024-01-15 | 0.1 | Début d’introduction | Omar Ben Thami |
| 2024-01-18 | 0.2 | Amélioration des paquets de communication | Omar Ben Thami |
| 2024-01-19 | 0.3 | Communication client serveur (WebSocket) | Omar Ben Thami |
| 2024-01-27 | 0.4 | Communication client serveur (HTTP) | Omar Ben Thami |
| 2024-01-28 | 0.5 | Finalisation des paquets | Omar Ben Thami |
| 2024-01-30 | 0.5 | Ajout d’un paquet manquant | Omar Ben Thami |
| 2024-01-30 | 1.0 | Révision du document et mise en page | Olivier Tremblay-Noël |

Table des matières

[**1. Introduction 4**](#_heading=h.30j0zll)

[**2. Communication client-serveur 4**](#_heading=h.1fob9te)

[**3. Description des paquets pour les sockets 5**](#_heading=h.3znysh7)

[3.1 Socket pour le Début de la Partie : 5](#_heading=h.ay138xckti2h)

[3.2 Socket pour l'Action en Jeu : 5](#_heading=h.mxf036fttlh6)

[3.3 Socket pour Abandonner une Partie : 5](#_heading=h.t2lzn3u7fq3l)

[3.4 Socket pour Valider une Différence Trouvée : 5](#_heading=h.k6wb2blc22gz)

[3.5 Socket pour l'Événement de Clic : 5](#_heading=h.qqwyfm97i020)

[3.6 Socket pour le Chronomètre de Jeu : 5](#_heading=h.4s58hm9g8mbb)

[3.7 Socket pour la Fin de la Partie : 5](#_heading=h.u1f3fobkxi58)

[3.8 Socket pour le Chat en Jeu : 5](#_heading=h.4n3wvku4ti0o)

[3.9 Socket pour Chat Général : 6](#_heading=h.ra4pijb8eep)

[3.10 Socket pour les Mises à Jour de la Partie : 6](#_heading=h.3eih6it269c3)

[3.11 Socket pour les Observateurs : 6](#_heading=h.1o53kv8s5tm9)

[3.12 Socket pour les Mises à Jour d'Observateur : 6](#_heading=h.1ultb8jbiu7e)

[**4. Description des paquets pour protocole HTTP: 6**](#_heading=h.jqim3pa6t08c)

[4.1 Système de compte - Authentification: 6](#_heading=h.oowjaswm802)

[4.1.1 Connexion: 6](#_heading=h.3idbxa1d4v8x)

[4.1.2 Création de compte: 7](#_heading=h.lxu5i12344e4)

[4.1.3 Déconnexion: 7](#_heading=h.jzithxt4f73y)

[4.1.4 Modification du profil: 8](#_heading=h.vltgv0tdrhv4)

[4.2 Intégration du clavardage: 8](#_heading=h.xla46szgpgok)

[4.2.1 Envoi d’un message: 8](#_heading=h.3uhf0gyz33ke)

[4.2.1 Historique des messages d’un canal: 9](#_heading=h.c8jpo8dvu2dq)

[4.3 Clavardage - Canaux de discussion: 9](#_heading=h.cj25ecdhbgkp)

[4.3.1 Création d’un nouveau canal: 9](#_heading=h.l74wkfpozmoc)

[4.3.2 Rejoindre un canal existant: 10](#_heading=h.yuy627k9k6az)

[4.3.3 Quitter un canal: 10](#_heading=h.m2zpf1akv8zj)

[4.4 Système - Jeu et Partie: 11](#_heading=h.t57cyotklv8p)

[4.4.1 Création d’une partie: 11](#_heading=h.bf6w08p19q9n)

[4.4.2 Rejoindre la salle d’attente d’une partie: 11](#_heading=h.h3j784pgr8qy)

[4.4.3 Commencer une partie: 12](#_heading=h.qnf4tpa8own)

[4.4.4 Supprimer une carte de jeu: 12](#_heading=h.1s24sk67m04a)

[4.4.5 Télécharger une carte de jeu: 13](#_heading=h.tfwoqpl8k7u6)

[4.4.6 Téléverser une carte de jeu: 13](#_heading=h.1yutnp67684)

[4.4.7 Historique de jeu: 14](#_heading=h.1tfqt1hdolqh)

[4.4.8 Suppression de l’historique de jeu: 14](#_heading=h.hs4d91yq38lx)

[4.5 Système - Boutique: 15](#_heading=h.fchtew6dfwj3)

[4.5.1 Acheter des items: 15](#_heading=h.ja9jd2549wbo)

[4.5.2 Personnaliser l’avatar: 16](#_heading=h.24zsuh76nrn9)

[4.5.3 Accéder à la boutique: 16](#_heading=h.id63y8q5fd9k)

Protocole de communication

# 1. Introduction

Dans ce document, nous détaillons le protocole de communication mis en place pour le projet *PolyDiff*. Notre objectif est d'élaborer une architecture robuste et efficiente, permettant une communication fluide et sécurisée entre le client et le serveur.

Ce protocole est le fondement de toutes les interactions au sein de notre application. Il couvre deux aspects principaux : la communication en temps réel via WebSockets, facilitée par la librairie Socket.io, et les échanges de données standard via les requêtes HTTP. Ensemble, ces mécanismes de communication jouent un rôle crucial dans le fonctionnement harmonieux de notre application, assurant à la fois réactivité et fiabilité.

Nous présentons ici les principes de base, les structures de données, et les différents cas d'utilisation qui guident notre approche de communication. Ce document est destiné à tous les développeurs et parties prenantes du projet, fournissant une référence claire et précise pour la mise en œuvre et la maintenance du système de communication.

# 

# 2. Communication client-serveur

Dans le cadre du développement de *PolyDiff*, nous avons opté pour une approche hybride en matière de communication client-serveur, utilisant à la fois la librairie Socket.io pour les WebSockets et des requêtes HTTP classiques. Cette combinaison nous permet de tirer parti des forces de chaque technologie pour offrir une expérience utilisateur fluide et réactive.

**Utilisation de Socket.io et WebSockets :**

La librairie Socket.io, s'appuyant sur le protocole WebSocket, joue un rôle clé dans notre architecture de communication. Elle nous permet d'établir une connexion bidirectionnelle persistante entre le client et le serveur. Cette connexion est cruciale pour le déroulement en temps réel du jeu, facilitant la mise à jour dynamique des clients, la gestion de la logique de jeu en mode multijoueur, et l'échange de messages instantanés entre les joueurs. Un exemple concret de l'utilisation de cette technologie est observé dans la gestion des parties multijoueurs. Les joueurs se connectent via des salles de jeu créées ou gérées par Socket.io. Chaque action dans le jeu, comme la découverte d'une différence ou la réalisation d'un mouvement, est communiquée instantanément à tous les participants, assurant ainsi une synchronisation parfaite entre eux. De plus, cette approche nous permet de notifier les clients de tout changement survenu sur le serveur sans qu'une requête préalable soit nécessaire. Par exemple, lors de l'ajout ou de la suppression d'une fiche de jeu, le serveur informe immédiatement tous les clients connectés.

**Utilisation des requêtes HTTP :**

En parallèle, les requêtes HTTP sont utilisées pour des interactions plus conventionnelles avec le serveur, telles que le chargement initial des fiches de jeu, la création de nouvelles fiches, ou la suppression de jeux existants. Ces requêtes suivent le modèle classique requête-réponse, où le client envoie une requête au serveur et attend sa réponse pour procéder. Ce mode de communication est particulièrement adapté pour des actions auxquelles la bidirectionnalité n'est pas nécessaire. Par exemple, lors de la création d'une nouvelle fiche de jeu, le client soumet une requête POST contenant les détails de la fiche. Le serveur traite cette requête, crée la fiche, et renvoie une réponse confirmant l'opération. Ce processus garantit la clarté, la sécurité et l'efficacité de la communication pour des tâches spécifiques.

En somme, notre choix de combiner les WebSockets via Socket.io avec des requêtes HTTP traditionnelles nous offre le meilleur des deux mondes : une communication en temps réel pour les aspects critiques du jeu et une interaction fiable et structurée pour les opérations standards. Cette approche hybride nous permet de répondre efficacement aux différents besoins de communication entre le client et le serveur.

# 3. Description des paquets pour les sockets

## 3.1 Socket pour le Début de la Partie :

Événement : 'gameStart'

Données : Identifiants de la partie, joueurs participants, paramètres de la partie.

Description : Déclenché lorsque le créateur de la partie lance le jeu.

## 3.2 Socket pour l'Action en Jeu :

Événement : 'gameAction'

Données : Type d'action (par exemple, trouver une différence), coordonnées ou détails de l'action.

Description : Utilisé pour communiquer les actions des joueurs en temps réel, comme cliquer sur une différence.

## 3.3 Socket pour Abandonner une Partie :

Événement : 'gameAbort'

Données : Identifiant du joueur, raison de l'abandon.

Description : Envoyé lorsqu'un joueur quitte la partie avant sa fin normale. Peut-être utilisé pour informer les autres joueurs de l'abandon et ajuster l'état du jeu en conséquence.

## 3.4 Socket pour Valider une Différence Trouvée :

Événement : 'differenceValidation'

Données : Identifiant du joueur, coordonnées ou détails de la différence trouvée.

Description : Envoyé lorsqu'un joueur prétend avoir trouvé une différence. Le serveur reçoit cet événement et valide si la différence est correcte ou non.

## 3.5 Socket pour l'Événement de Clic :

Événement : 'gameClick'

Données : Identifiant du joueur, position du clic (par exemple, coordonnées x, y), identifiant de la partie.

Description : Cet événement est déclenché lorsque le joueur clique sur le jeu. La position du clic est envoyée au serveur, qui la compare avec les emplacements des différences connues dans le tableau des différences.

## 3.6 Socket pour le Chronomètre de Jeu :

Événement : 'timerUpdate'

Données : Temps restant.

Description : Mise à jour périodique du temps restant dans le jeu.

## 3.7 Socket pour la Fin de la Partie :

Événement : 'gameEnd'

Données : Résultats finaux, scores des joueurs.

Description : Envoyé lorsque la partie est terminée, avec les résultats et les scores.

## 3.8 Socket pour le Chat en Jeu :

Événement : 'gameChatMessage'

Données : Message, expéditeur.

Description : Permet aux joueurs de communiquer dans le chat dédié à la partie en cours.

## 3.9 Socket pour Chat Général :

Événement : 'gameChatMessage'

Données : Message, expéditeur, salle (optionnellement, si hors-jeu).

Description : Permet la communication entre les joueurs dans un chat global. Si la salle n'est pas spécifiée, le message est considéré comme appartenant au chat général. Si une salle est spécifiée, le message est envoyé uniquement aux joueurs dans cette salle, ce qui est utile pour le chat en jeu.

## 3.10 Socket pour les Mises à Jour de la Partie :

Événement : 'gameUpdate'

Données : Mises à jour de l'état du jeu (par exemple, changements de score, actions des joueurs).

Description : Pour informer tous les joueurs des mises à jour importantes sur l'état de la partie.

## 3.11 Socket pour les Observateurs :

Événement : 'observerJoin'

Données : Identifiant de l'observateur, partie observée.

Description : Utilisé lorsque quelqu'un rejoint la partie en tant qu'observateur.

## 3.12 Socket pour les Mises à Jour d'Observateur :

Événement : 'observerUpdate'

Données : Actions ou événements spécifiques à observer.

Description : Envoyé pour tenir les observateurs informés des événements en cours dans la partie.

# 4. Description des paquets pour protocole HTTP:

## 4.1 Système de compte - Authentification:

### 4.1.1 Connexion:

| **Attribut** | **Valeur** | **Description** |
| --- | --- | --- |
| URL | api/fs/players/login |  |
| Type de requête | POST |
| **Request body** | | |
| En-tête de demande | Content-Type: application/json | Indique que le corps de la requête est en JSON. |
| Corps de la requête | {"username": "string", "password": "string"} | Identifiants de l'utilisateur pour la connexion. |
| **VALIDE** | | |
| Status | 200 | Connexion réussie. |
| **INVALIDE** | | |
| Status | 400 | Données invalides dans la requête. |
| 401 | Échec d'authentification, mauvais nom d'utilisateur ou mot de passe. |

### 4.1.2 Création de compte:

| **Attribut** | **Valeur** | **Description** |
| --- | --- | --- |
| URL | api/fs/players/new |  |
| Type de requête | POST |
| **Information de requête** | | |
| En-tête de demande | Content-Type: application/json | Indique que le corps de la requête est en JSON. |
| Corps de la requête | {"username": "string", "email": "string", "password": "string"} | Données de l'utilisateur pour créer un compte. |
| **VALIDE** | | |
| Status | 201 | Compte créé avec succès. |
| **INVALIDE** | | |
| Status | 400 | Données invalides dans la requête. |
| 409 | Conflit (nom d’utilisateur déjà utilisé) |

### 4.1.3 Déconnexion:

| **Attribut** | **Valeur** | **Description** |
| --- | --- | --- |
| URL | api/fs/players/:username/logout |  |
| Type de requête | PATCH |
| **Informations de requête** | | |
| En-tête de demande | Authorization: Bearer <token> | Header pour le token d'authentification de l'utilisateur. |
| Content-Type: application/json | Indique que le corps de la requête est en JSON. |
| **VALIDE** | | |
| Status | 200 | Utilisateur a été déconnecté. |
| **INVALIDE** | | |
| Status | 401 | Token d'authentification invalide ou absent. |

### 

### 4.1.4 Modification du profil:

| **Attribut** | **Valeur** | **Description** |
| --- | --- | --- |
| URL | api/fs/players/:username/profile |  |
| Type de requête | PUT |
| **Information de requête** | | |
| En-tête de demande | Content-Type: application/json | Indique le type de média des ressources |
| Authorization: Bearer token | Token d'authentification de l'utilisateur. |
| Corps de la requête | { "new\_username": "string", "avatar": "url" } | JSON contenant le nouveau pseudonyme et l'URL de l'avatar. |
| **VALIDE** | | |
| Status | 200 | Profil mis à jour avec succès. |
| **INVALIDE** | | |
| Status | 400 | Requête invalide, informations manquantes ou mal formées. |

## 4.2 Intégration du clavardage:

### 4.2.1 Envoi d’un message:

| **Attribut** | **Valeur** | **Description** |
| --- | --- | --- |
| URL | api/fs/chat/:channelId/message |  |
| Type de requête | POST |
| **Information de requête** | | |
| En-tête de demande | Content-Type: application/json | Indique le type de média des ressources |
| Authorization: Bearer token | Token d'authentification de l'utilisateur. |
| Corps de la requête | { "message": "string" } | Contenu du message à envoyer. |
| **VALIDE** | | |
| Status | 201 | Message envoyé avec succès. |
| **INVALIDE** | | |
| Status | 400 | Données invalides dans la requête. |
| 401 | Échec d'authentification, token invalide ou absent. |
| 404 | Canal de chat non trouvé. |

## 

### 4.2.1 Historique des messages d’un canal:

| **Attribut** | **Valeur** | **Description** |
| --- | --- | --- |
| URL | api/fs/chat/:channelId/history |  |
| Type de requête | GET |
| **Information de requête** | | |
| En-tête de demande | Authorization: Bearer token | Token d'authentification de l'utilisateur. |
| **VALIDE** | | |
| Status | 200 | Historique des messages récupéré avec succès. |
| **INVALIDE** | | |
| Status | 401 | Échec d'authentification, token invalide ou absent. |
| 404 | Canal de chat non trouvé. |

## 

## 4.3 Clavardage - Canaux de discussion:

### 4.3.1 Création d’un nouveau canal:

| **Attribut** | **Valeur** | **Description** |
| --- | --- | --- |
| URL | api/fs/chat/channels/new |  |
| Type de requête | POST |
| **Information de requête** | | |
| En-tête de demande | Content-Type: application/json | Indique le type de média des ressources |
| Authorization: Bearer token | Token d'authentification de l'utilisateur. |
| Corps de la requête | { "channel\_name": "string", "public": "boolean" } | Informations pour la création du canal. |
| **VALIDE** | | |
| Status | 201 | Nouveau canal créé avec succès. |
| **INVALIDE** | | |
| Status | 400 | Données invalides dans la requête. |
| 401 | Échec d'authentification, token invalide ou absent. |

## 

### 4.3.2 Rejoindre un canal existant:

| **Attribut** | **Valeur** | **Description** |
| --- | --- | --- |
| URL | api/fs/chat/channels/:channelId/join |  |
| Type de requête | POST |
| **Information de requête** | | |
| En-tête de demande | Authorization: Bearer token | Token d'authentification de l'utilisateur. |
| **VALIDE** | | |
| Status | 200 | Rejoindre le canal avec succès. |
| **INVALIDE** | | |
| Status | 400 | Données invalides dans la requête. |
| 401 | Échec d'authentification, token invalide ou absent. |
| 404 | Canal de chat non trouvé. |

## 

### 4.3.3 Quitter un canal:

| **Attribut** | **Valeur** | **Description** |
| --- | --- | --- |
| URL | api/fs/chat/channels/:channelId |  |
| Type de requête | DELETE |
| **Information de requête** | | |
| En-tête de demande | Authorization: Bearer token | Token d'authentification de l'utilisateur. |
| **VALIDE** | | |
| Status | 200 | Utilisateur a quitté le canal avec succès. |
| **INVALIDE** | | |
| Status | 400 | Données invalides dans la requête. |
| 401 | Échec d'authentification, token invalide ou absent. |
| 404 | Canal de chat non trouvé. |

## 

## 4.4 Système - Jeu et Partie:

### 4.4.1 Création d’une partie:

| **Attribut** | **Valeur** | **Description** |
| --- | --- | --- |
| URL | api/fs/games/new |  |
| Type de requête | POST |
| **Information de requête** | | |
| En-tête de demande | Content-Type: application/json | Indique le type de média des ressources |
| Authorization: Bearer token | Token d'authentification de l'utilisateur. |
| Corps de la requête | { "game\_type": "string", "settings": {...} } | Détails et préférences pour la nouvelle partie. |
| **VALIDE** | | |
| Status | 201 | Partie créée avec succès. |
| **INVALIDE** | | |
| Status | 400 | Données invalides dans la requête. |
| 401 | Échec d'authentification, token invalide ou absent. |

## 

### 4.4.2 Rejoindre la salle d’attente d’une partie:

| **Attribut** | **Valeur** | **Description** |
| --- | --- | --- |
| URL | api/fs/games/:gameId/join |  |
| Type de requête | POST |
| **Information de requête** | | |
| En-tête de demande | Authorization: Bearer token | Token d'authentification de l'utilisateur. |
| **VALIDE** | | |
| Status | 200 | Rejoint la salle d'attente avec succès. |
| **INVALIDE** | | |
| Status | 400 | Données invalides dans la requête. |
| 401 | Échec d'authentification, token invalide ou absent. |
| 404 | Partie non trouvée. |

### 4.4.3 Commencer une partie:

| **Attribut** | **Valeur** | **Description** |
| --- | --- | --- |
| URL | api/fs/games/:gameId/start |  |
| Type de requête | POST |
| **Information de requête** | | |
| En-tête de demande | Authorization: Bearer token | Token d'authentification de l'utilisateur. |
| **VALIDE** | | |
| Status | 200 | Partie commencée avec succès. |
| **INVALIDE** | | |
| Status | 400 | Données invalides dans la requête. |
| 401 | Échec d'authentification, token invalide ou absent. |
| 403 | Utilisateur non autorisé à commencer la partie. |
| 404 | Partie non trouvée. |

### 4.4.4 Supprimer une carte de jeu:

| **Attribut** | **Valeur** | **Description** |
| --- | --- | --- |
| URL | api/fs/gamecards/:cardId |  |
| Type de requête | DELETE |
| **Information de requête** | | |
| En-tête de demande | Authorization: Bearer token | Token d'authentification de l'utilisateur. |
| **VALIDE** | | |
| Status | 200 | Game card supprimée avec succès. |
| **INVALIDE** | | |
| Status | 401 | Échec d'authentification, token invalide ou absent. |
| 403 | Utilisateur non autorisé à supprimer la game card. |
| 404 | Game card non trouvée. |

## 

### 4.4.5 Télécharger une carte de jeu:

| **Attribut** | **Valeur** | **Description** |
| --- | --- | --- |
| URL | api/fs/gamecards/:cardId/download |  |
| Type de requête | GET |
| **Information de requête** | | |
| En-tête de demande | Authorization: Bearer token | Token d'authentification de l'utilisateur. |
| **VALIDE** | | |
| Status | 200 | Game card téléchargée avec succès. |
| **INVALIDE** | | |
| Status | 401 | Échec d'authentification, token invalide ou absent. |
| 404 | Game card non trouvée. |

### 

### 4.4.6 Téléverser une carte de jeu:

| **Attribut** | **Valeur** | **Description** |
| --- | --- | --- |
| URL | api/fs/gamecards/new |  |
| Type de requête | POST |
| **Information de requête** | | |
| En-tête de demande | Authorization: Bearer token | Token d'authentification de l'utilisateur. |
| Content-Type: multipart/form-data | Indique que le corps de la requête contient des fichiers. |
| Corps de la requête | FormData avec fichier de la game card | Les données du fichier de la game card à téléverser. |
| **VALIDE** | | |
| Status | 201 | Game card téléversée avec succès. |
| **INVALIDE** | | |
| Status | 400 | Données invalides dans la requête. |
| 401 | Échec d'authentification, token invalide ou absent. |
| 413 | Fichier trop grand. |

### 

### 4.4.7 Historique de jeu:

| **Attribut** | **Valeur** | **Description** |
| --- | --- | --- |
| URL | api/fs/games/history |  |
| Type de requête | GET |
| **Information de requête** | | |
| En-tête de demande | Authorization: Bearer token | Token d'authentification de l'utilisateur. |
| **VALIDE** | | |
| Status | 200 | Historique des jeux récupéré avec succès. |
| **INVALIDE** | | |
| Status | 401 | Échec d'authentification, token invalide ou absent. |
| 404 | Historique de jeu non trouvé ou utilisateur n'a pas d'historique. |

## 

### 4.4.8 Suppression de l’historique de jeu:

| **Attribut** | **Valeur** | **Description** |
| --- | --- | --- |
| URL | api/fs/games/history/:historyId |  |
| Type de requête | DELETE |
| **Information de requête** | | |
| En-tête de demande | Authorization: Bearer token | Token d'authentification de l'utilisateur. |
| **VALIDE** | | |
| Status | 200 | Enregistrement de l'historique de jeu supprimé avec succès. |
| **INVALIDE** | | |
| Status | 401 | Échec d'authentification, token invalide ou absent. |
| 403 | Utilisateur non autorisé à supprimer l'historique. |
| 404 | Enregistrement de l'historique non trouvé. |

## 

## 4.5 Système - Boutique:

### 4.5.1 Acheter des items:

| **Attribut** | **Valeur** | **Description** |
| --- | --- | --- |
| URL | api/fs/shop/items/buy |  |
| Type de requête | POST |
| **Information de requête** | | |
| En-tête de demande | Content-Type: application/json | Indique le type de média des ressources |
| Authorization: Bearer token | Token d'authentification de l'utilisateur. |
| Corps de la requête | { "item\_id": "string", "quantity": "number" } | Identifiant de l'item et quantité à acheter. |
| **VALIDE** | | |
| Status | 200 | Achat réalisé avec succès. |
| **INVALIDE** | | |
| Status | 400 | Données invalides dans la requête ou solde insuffisant. |
| 401 | Échec d'authentification, token invalide ou absent. |
| 404 | Item non trouvé ou indisponible. |

### 4.5.2 Personnaliser l’avatar:

| **Attribut** | **Valeur** | **Description** |
| --- | --- | --- |
| URL | api/fs/players/:username/customize |  |
| Type de requête | POST |
| **Information de requête** | | |
| En-tête de demande | Content-Type: application/json | Indique le type de média des ressources |
| Authorization: Bearer token | Token d'authentification de l'utilisateur. |
| Corps de la requête | { "avatar\_settings": {...} } | Paramètres de personnalisation de l'avatar. |
| **VALIDE** | | |
| Status | 200 | Avatar personnalisé avec succès. |
| **INVALIDE** | | |
| Status | 400 | Données invalides dans la requête ou solde insuffisant. |
| 401 | Échec d'authentification, token invalide ou absent. |

## 

### 4.5.3 Accéder à la boutique:

| **Attribut** | **Valeur** | **Description** |
| --- | --- | --- |
| URL | api/fs/shop/items |  |
| Type de requête | GET |
| **Information de requête** | | |
| En-tête de demande | Authorization: Bearer token | Token d'authentification de l'utilisateur. |
| **VALIDE** | | |
| Status | 200 | Liste des items récupérée avec succès. |
| **INVALIDE** | | |
| Status | 401 | Échec d'authentification, token invalide ou absent. |

## 