**Règles d’utilisation des webservices**

Ce document recense les différents cas de gestion possibles des webservices selon les situations.

# UserController

## Règles pour la méthode getUser (GET /api/users/{id}) :

* L'endpoint permet de récupérer un utilisateur par son ID.
* L'ID de l'utilisateur doit être fourni en tant que chemin de l'URL.
* Si l'utilisateur n'est pas trouvé (UserNotFoundException), renvoie une réponse HTTP avec le statut NO\_CONTENT.

## Règles pour la méthode saveUser (POST /api/users) :

* L'endpoint permet de créer un nouvel utilisateur.
* Les données de l'utilisateur doivent être fournies dans le corps de la requête au format JSON.
* Si la création de l'utilisateur est réussie, renvoie l'utilisateur créé en réponse.

## Règles pour la méthode deleteUser (DELETE /api/users/{id}) :

* L'endpoint permet de supprimer un utilisateur par son ID.
* L'ID de l'utilisateur à supprimer doit être fourni en tant que chemin de l'URL.
* Si l'utilisateur n'est pas trouvé (UserNotFoundException), renvoie une réponse HTTP avec le statut BAD\_REQUEST.

# PrefabController

## Règles pour la méthode getAllPrefabs (GET /api/prefabs) :

* L'endpoint permet de récupérer la liste de tous les prefabs dans la base de données.
* Si la liste de prefabs est vide, renvoie une réponse HTTP avec le statut NO\_CONTENT.

## Règles pour la méthode getPrefab (GET /api/prefabs/{name}) :

* L'endpoint permet de récupérer un prefab par son nom.
* Le nom du prefab doit être fourni en tant que chemin de l'URL.
* Si le prefab n'est pas trouvé (un prefab par défaut est renvoyé), renvoie une réponse HTTP avec le statut NO\_CONTENT.

## Règles pour la méthode postPrefab (POST /api/prefabs) :

* L'endpoint permet d'ajouter un nouveau prefab à moins qu'un prefab du même nom n'existe déjà.
* Les données du nouveau prefab doivent être fournies dans le corps de la requête au format JSON.
* En cas de succès, aucune réponse n'est renvoyée, le statut de réponse est OK (200).
* En cas d'échec (InvalidPrefabException ou ExistingPrefabException), renvoie une réponse HTTP avec le statut BAD\_REQUEST.

## Règles pour la méthode patchPrefab (PATCH/PUT /api/prefabs/{name}) :

* L'endpoint permet de mettre à jour un prefab existant par son nom.
* Le nom du prefab à mettre à jour doit être fourni en tant que chemin de l'URL.
* Les données mises à jour du prefab doivent être fournies dans le corps de la requête au format JSON.
* En cas de succès, aucune réponse n'est renvoyée, le statut de réponse est OK (200).
* En cas d'échec (NoExistingPrefabException, PrefabNameNotMatchingException, ou InvalidPrefabException), renvoie une réponse HTTP avec le statut BAD\_REQUEST.

## Règles pour la méthode deletePrefab (DELETE /api/prefabs/{name}) :

* L'endpoint permet de supprimer un prefab par son nom.
* Le nom du prefab à supprimer doit être fourni en tant que chemin de l'URL.
* En cas de succès, aucune réponse n'est renvoyée, le statut de réponse est OK (200).
* En cas d'échec (NoExistingPrefabException), renvoie une réponse HTTP avec le statut BAD\_REQUEST.

# FormDataController

## Règles pour la méthode getAllFormData (GET /api/formdata) :

* L'endpoint permet de récupérer la liste de toutes les données de formulaire dans la base de données.

## Règles pour la méthode getAllFormDataFromPrefab (GET /api/formdata/{prefab}) :

* L'endpoint permet de récupérer toutes les données de formulaire liées à un prefab spécifié.
* Le nom du prefab doit être fourni en tant que chemin de l'URL.

## Règles pour la méthode getAllFormDataFromPrefabField (GET /api/formdata/{prefab}/{group}/{field}) :

* L'endpoint permet de récupérer toutes les valeurs de champ liées à une combinaison spécifique de prefab, groupe et champ.
* Les noms du prefab, du groupe et du champ doivent être fournis en tant que chemins de l'URL.
* Si aucune donnée n'est trouvée, renvoie une réponse HTTP avec le statut NO\_CONTENT.

## Règles pour la méthode postFormData (POST /api/formdata/{prefab}) :

* L'endpoint permet d'ajouter de nouvelles données de formulaire liées à un prefab spécifié.
* Les données de formulaire doivent être fournies dans le corps de la requête au format JSON.
* En cas de succès, aucune réponse n'est renvoyée, le statut de réponse est OK (200).
* En cas d'échec (InvalidFormDataException), renvoie une réponse HTTP avec le statut BAD\_REQUEST.

Règles pour la méthode patchFormData (PATCH/PUT /api/formdata/{prefab}) :

* L'endpoint permet de mettre à jour des données de formulaire existantes liées à un prefab spécifié.
* Les données de formulaire mises à jour doivent être fournies dans le corps de la requête au format JSON.
* En cas de succès, aucune réponse n'est renvoyée, le statut de réponse est OK (200).
* En cas d'échec (NoExistingFormDataException, InvalidFormDataException), renvoie une réponse HTTP avec le statut BAD\_REQUEST.

## Règles pour la méthode deleteFormData (DELETE /api/formdata) :

* L'endpoint permet de supprimer des données de formulaire en utilisant l'identifiant de l'objet FormData.
* L'identifiant de l'objet FormData doit être fourni dans le corps de la requête au format JSON.
* En cas de succès, aucune réponse n'est renvoyée, le statut de réponse est OK (200).
* En cas d'échec (NoExistingFormDataException), renvoie une réponse HTTP avec le statut BAD\_REQUEST.