Spécifications et Documentation

API REST

Mise en place d'un service REST qui permet d'accéder aux données concernant des bibliothèques et les livres qu'elles contiennent.

Les Livres (Book)

Méthode	GET
Une description fonctionnelle incluant les contraintes éventuelles	Il y a 2 méthodes GET, la première permet de retourner tous les livres et leurs détails enregistrés dans la base de données , la deuxième affiche les détails d'un livre dont l'on aura spécifié l'id dans l'URL.
L'URL exposée pour le service en question	Pour Retourner tous les livres : http://localhost:8080/api/book ou http://localhost:8080/api/books Pour retourner un seul livre avec son id : http://localhost:8080/api/book/$id

La structure des données en réponse du service	En JSON, par exemple :
	{
	"id":1,
	"library":{"id":1},
	"name":"Le rouge et le noir",
	"releaseDate":"1906-02-03T23:00:00Z","
	isbn":"76GFRY75",
	"author":"Stendhal"
	}

Un listing des cas d'erreurs gérés accompagnés d'une brève description	Retourne un erreur 404 si le livre n'existe pas, si l'id ne correspond à aucun livre de la base de donnée.

Méthode	POST
Description fonctionnelle incluant les contraintes éventuelles	Méthode permettant de créer une nouvelle instance de book dans la base de données. Il faut préciser tous les attributs pour pouvoir créer une nouvelle entité de livre . La date de sortie doit être entré dans ce format : YYYY-MM-DDTheure:minute:secondeZ"
L'URL exposée pour le service en question	http://127.0.0.1:8080/api/book/?name=$name&author=$author&releaseDate=$releaseDate&isbn=$isbn&library.id=$id Les variables commençant par "$" doivent être remplacé par la valeur que l'on souhaite leur attribuer.

La structure des données en réponse du service	En JSON, par exemple :
	{
	"id":1,
	"library":{"id":1},
	"name":"Le rouge et le noir",
	"releaseDate":"1906-02-03T23:00:00Z","
	isbn":"76GFRY75",
	"author":"Stendhal"
	}

Un listing des cas d'erreurs gérés accompagnés d'une brève description	Retourne un erreur 404 si le la bibliothèque préciser en paramètre n'existe pas . Une erreur 400 si la requête n'est pas formuler correctement , par exemple si la date n'est pas dans le bon format ou si manque un attribut censé être non null.

Méthode	PUT
Description fonctionnelle incluant les contraintes éventuelles	Méthode permettant de modifier un livre existant déjà dans la base de données. Elle permet de modifier un ou plusieurs attribut à la fois.
L'URL exposée pour le service en question	http://127.0.0.1:8080/api/book/$id/?name=$name&author=$author&releaseDate=$releaseDate&isbn=$isbn&library.id=$id Les variables commençant par "$" doivent être remplacé par la valeur que l'on souhaite leur attribuer. Toutefois, tous les attributs ne sont pas nécessaire, on ne rentre dans l'url uniquement ceux que l'on souhaite modifier.

La structure des données en réponse du service	En JSON, par exemple :
	{
	"id":1,
	"library":{"id":1},
	"name":"Le rouge et le noir",
	"releaseDate":"1906-02-03T23:00:00Z","
	isbn":"76GFRY75",
	"author":"Stendhal"
	}
	
	Avec les modifications souhaitées mises à jour.

Un listing des cas d'erreurs gérés accompagnés d'une brève description	Retourne un erreur 404 si le livre que l'on souhaite modifier n'existe pas . Une erreur 400 si l'attribut n'est pas dans le bon format, exemple la date n'est pas mise au format YYYY-MM-DDTheure:minute:secondeZ .

Méthode	DELETE
Description fonctionnelle incluant les contraintes éventuelles	Méthode permettant de supprimer un livre existant dans la base de données.
L'URL exposée pour le service en question	http://127.0.0.1:8080/api/book/$id Supprime le livre correspondant à l'id "$id"
La structure des données en réponse du service	Le service ne renvoie pas de données vu qu'il s'agit d'une suppression, uniquement une phrase : "Le livre ${params.id} a été Supprimé." ou : "Le livre ${params.id} est inexistant." en cas d'erreur
Un listing des cas d'erreurs gérés accompagnés d'une brève description	Retourne un erreur 404 si le livre que l'on souhaite supprimer n'existe pas .

Les Bibliothèques (Library)

Méthode	GET
Une description fonctionnelle incluant les contraintes éventuelles	Il y a 2 méthodes GET, la première permet de retourner toutes les bibliothèques et leurs détails enregistrées dans la base de données , la deuxième affiche les détails d'un livre dont l'on aura spécifier l'id dans l'URL.
L'URL exposée pour le service en question	Pour Retourner toutes les bibliothèques : http://localhost:8080/api/library ou http://localhost:8080/api/libraries Pour retourner une seule bibliothèque avec son id : http://localhost:8080/api/library/$id

La structure des données en réponse du service	En JSON, par exemple :
	{
	"id": 1,
	"address": "2 Place Yves Klein, 06300 Nice",
	"books": [
	{
	"id": 1
	},
	{
	"id": 2
	}
	],
	"name": "Louis Nucera",
	"yearCreated": 2002
	}

Un listing des cas d'erreurs gérés accompagnés d'une brève description	Retourne un erreur 404 si la bibliothèque n'existe pas, si l'id ne correspond à aucun livre de la base de donnée.

Méthode	POST
Description fonctionnelle incluant les contraintes éventuelles	Méthode permettant de créer une nouvelle instance de library dans la base de données. Il faut préciser tous les champs pour pouvoir créer une nouvelle entité de bibliothèque.
L'URL exposée pour le service en question	http://127.0.0.1:8080/api/library/?name=$name&yearCreated=$yearCreated&address=$address Les variables commençant par "$" doivent être remplacé par la valeur que l'on souhaite leur attribuer.

La structure des données en réponse du service	En JSON, par exemple :
	[
	{
	"id": 3,
	"address": "test",
	"books": null,
	"name": "test",
	"yearCreated": 2017
	}
	]

Un listing des cas d'erreurs gérés accompagnés d'une brève description	Une erreur 400 si la requête n'est pas formuler correctement , par exemple si il manque un attribut censé être non null.

Méthode	PUT
Description fonctionnelle incluant les contraintes éventuelles	Méthode permettant de modifier une bibliothèque existant déjà dans la base de données. Elle permet de modifier un ou plusieurs attributs à la fois.
L'URL exposée pour le service en question	http://127.0.0.1:8080/api/library/$id/?name=$name&yearCreated=$yearCreated&address=$address Les variables commençant par "$" doivent être remplacé par la valeur que l'on souhaite leur attribuer. Toutefois, tous les attributs ne sont pas nécessaire, on ne rentre dans L'URL uniquement ceux que l'on souhaite modifier.

La structure des données en réponse du service	En JSON, par exemple :
	{
	"id": 1,
	"address": "2 Place Yves Klein, 06300 Nice",
	"books": [
	{
	"id": 1
	},
	{
	"id": 2
	}
	],
	"name": "Louis Nucera",
	"yearCreated": 2002
	}
	Avec les modifications souhaitées mises à jour.

Un listing des cas d'erreurs gérés accompagnés d'une brève description	Retourne un erreur 404 si la bibliothèque que l'on souhaite modifier n'existe pas . Une erreur 400 si l'attribut n'est pas dans le bon format, par exemple "yearCreated" doit être un Integer et non une chaîne de caractères.

Méthode	DELETE
Description fonctionnelle incluant les contraintes éventuelles	Méthode permettant de supprimer une bibliothèque existant dans la base de données.
L'URL exposée pour le service en question	http://127.0.0.1:8080/api/library/$id Supprime le livre correspondant à l'id "$id"
La structure des données en réponse du service	Le service ne renvoie pas de données vu qu'il s'agit d'une suppression, uniquement une phrase : "La bibliothèque ${params.id} a été supprimé" ou : "La bibliothèque ${params.id} est inexistante." en cas d'erreur
Un listing des cas d'erreurs gérés accompagnés d'une brève description	Retourne un erreur 404 si la bibliothèque que l'on souhaite supprimer n'existe pas .

Les Ressources liées

Méthode	GET
Une description fonctionnelle incluant les contraintes éventuelles	Il y a 2 méthodes GET, la première permet de retourner toutes les livres existant dans une bibliothèque précise,dont l'id sera spécifié dans l'URL. la deuxième affiche les détails d'un seul livre dont l'on aura spécifier l'id du livre en plus dans l'URL.
L'URL exposée pour le service en question	Pour Retourner toutes les bibliothèques : http://127.0.0.1:8080/api/biblio/$id/livres Pour retourner une seule bibliothèque avec son id : http://localhost:8080/api/biblio/$id/livres/?book.id=$bookid

La structure des données en réponse du service	En JSON, par exemple :
	[
	{
	"id": 2,
	"library": {
	"id": 1
	},
	"name": "les MisÃ©rables",
	"releaseDate": "1906-03-07T23:00:00Z",
	"isbn": "r657G78G",
	"author": "Victor Hugo"
	},
	{
	"id": 1,
	"library": {
	"id": 1
	},
	"name": "Le rouge et le noir",
	"releaseDate": "1906-02-03T23:00:00Z",
	"isbn": "76GFRY75",
	"author": "Stendhal"
	}
	]

Un listing des cas d'erreurs gérés accompagnés d'une brève description	Retourne une erreur 404 si la bibliothèque ou le livre n'existe pas ou si le livre existe mais pas dans la bibliothèque que l'on a spécifié.

Méthode	POST
Description fonctionnelle incluant les contraintes éventuelles	Méthode permettant de créer une nouvelle instance de livre à l'intérieur d'une bibliothèque précise dont l'id est spécifié dans l'URL . Il faut préciser tous les attributs nécessaires pour pouvoir créer une nouvelle entité de livre.
L'URL exposée pour le service en question	http://127.0.0.1:8080/api/biblio/$id/livres/?name=$name&author=$author&releaseDate=$releaseDate&isbn=$isbn Les variables commençant par "$" doivent être remplacé par la valeur que l'on souhaite leur attribuer.

La structure des données en réponse du service	En JSON, par exemple :
	[
	{
	"id": 3,
	"address": "test",
	"books": null,
	"name": "test",
	"yearCreated": 2017
	}
	]

Un listing des cas d'erreurs gérés accompagnés d'une brève description	Une erreur 400 si la requête n'est pas formuler correctement , par exemple si il manque un attribut censé être non null ou si un format ne correspond pas. Une erreur 404 si la bibliothèque n'existe pas.

Méthode	PUT
Description fonctionnelle incluant les contraintes éventuelles	Méthode permettant de modifier un livre dans la bibliothèque spécifié dans l'URL. Elle permet de modifier un ou plusieurs attributs à la fois.
L'URL exposée pour le service en question	http://127.0.0.1:8080/api/biblio/$id/livres/?book.id=$bookid&name=$name&author=$author&releaseDate=$releaseDate&isbn=$isbn Les variables commençant par "$" doivent être remplacé par la valeur que l'on souhaite leur attribuer. Toutefois, tous les attributs ne sont pas nécessaire, on ne rentre dans L'URL uniquement ceux que l'on souhaite modifier.

La structure des données en réponse du service	En JSON, par exemple :
	{
	"id": 1,
	"address": "2 Place Yves Klein, 06300 Nice",
	"books": [
	{
	"id": 1
	},
	{
	"id": 2
	}
	],
	"name": "Louis Nucera",
	"yearCreated": 2002
	}
	Avec les modifications souhaitées mises à jour.

Un listing des cas d'erreurs gérés accompagnés d'une brève description	Retourne un erreur 404 si la bibliothèque ou le livre n'existe pas ou si le livre existe mais pas dans la bibliothèque que l'on a spécifié. Une erreur 400 si l'attribut n'est pas dans le bon format.

Méthode	DELETE
Description fonctionnelle incluant les contraintes éventuelles	Méthode permettant de supprimer une bibliothèque existant dans la base de données.
L'URL exposée pour le service en question	http://localhost:8080/api/biblio/$id/livres/book.id=$bookid Supprime le livre correspondant à l'id "$bookid"
La structure des données en réponse du service	Le service ne renvoie pas de données vu qu'il s'agit d'une suppression, uniquement une phrase : "La bibliothèque ${params.id} a été supprimé" ou : "La bibliothèque ${params.id} est inexistante." en cas d'erreur
Un listing des cas d'erreurs gérés accompagnés d'une brève description	Retourne un erreur 404 si le livre ou la bibliothèque que l'on souhaite supprimer n'existe pas ou si le livre existe mais pas dans la bibliothèque spécifiée.