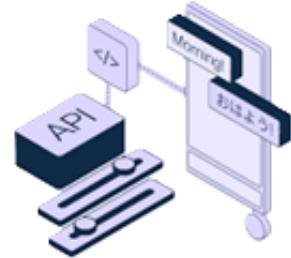


EXERCICE PRATIQUE

CONSOMMATION D'API : POST, PUT, DELETE

EXERCICE A REALISER SEUL

 BALISE IAG :  LIMITÉ : Autorisé uniquement pour assister l'apprentissage.



Dans le cours d'aujourd'hui, nous allons manipuler les données d'une API REST en utilisant les méthodes **POST**, **PUT** et **DELETE**. Plutôt que de chercher une API publique limitée sur Internet, nous allons créer en quelques minutes notre propre API **fonctionnelle** et **persistante**. Pour ce faire, nous utiliserons la solution **Beeceptor**, qui offre des avantages pédagogiques majeurs :

- Une persistance réelle contrairement à certaines API publiques
- Un CRUD automatique : il génère toutes les routes nécessaires GET, POST, PUT et DELETE.
- Zéro Code : Aucune programmation serveur n'est requise.
- Gratuit et instantané : la solution est prête à l'emploi sans clé d'API.

ÉTAPE 1 : CREATION DU SERVEUR

1. Rendez-vous sur beeceptor.com.
2. Dans le champ "**project-name**", saisissez un nom unique (ex: **service-de-donnees-votrePrénom**).
Note : Évitez les espaces et les caractères spéciaux.
3. Cliquez sur **Create Mock Server**.

Create your mock server

project-name

.free.beeceptor.com

Launch a mock server on a unique subdomain and simulate API behavior instantly.

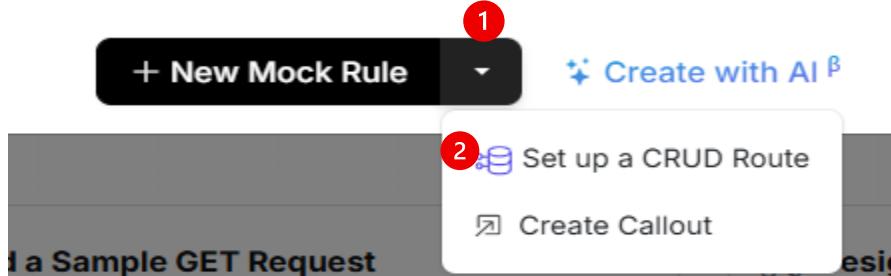


ÉTAPE 2 : CRÉATION DES ENDPOINTS

1. Sur la page affichée après la création du serveur Mock, cliquez sur « **Mocking Rules** » (en haut à droite)



2. Sur l'assistant qui s'ouvre, cliquez sur la liste déroulante à côté de « **New Mock Rule** » puis sélectionnez « **Set up a CRUD Route** »



3. Sur l'assistant qui s'ouvre, modifiez le champs « **API Path** » en remplaçant « **users** » par « **event** »

The screenshot shows two rows of configuration fields. The first row has an 'API Path' field containing '/api/users', an 'Identifier Field Name' field containing 'id', and a 'Manage Data' button. The second row shows the same fields after modification: 'API Path' contains '/api/event', 'Identifier Field Name' contains 'id', and 'Manage Data' is present. A large green arrow points downwards from the first row to the second row, indicating the change.

4. Cliquez sur « **Save Rule** ». Votre API est désormais prête à recevoir des requêtes via ses différentes routes.

ÉTAPE 3 : TEST ET VALIDATION

Utilisez Postman pour tester votre nouvelle API.

1. Vérification initiale (liste vide)

- **Requête :** **GET** <https://votre-sous-domaine.free.beceptor.com/event>
- **Résultat attendu :** [] (Code 200 OK)

2. Ajout d'un événement

- **Requête :** **POST** <https://votre-sous-domaine.free.beceptor.com/event>
- **Headers:** Content-Type: application/json (*juste une vérification*)
- **Body (JSON):**

```
{  
    "id": "1",  
    "titre": "Festival d'Hiver Saguenay ",  
    "date": "2026-02-20T20:00:00",  
    "lieu": "Zone Portuaire",  
    "artiste": "Orchestre Symphonique du CEGEP",  
    "prix": "35-85$"  
}
```

3. Vérification de la persistance

- **Requête :** **GET** <https://votre-sous-domaine.free.beceptor.com/event>
- **Résultat attendu :** Les données de votre festival s'affichent. (Code 200 OK)

4. Utilisez le fichier « **Evenements.json** » pour insérer des données supplémentaires dans votre API.

ÉTAPE 4 : TEST DE L'API AVEC C#

1. Ouvrez la solution C# fournie.
2. Configurez l'URL dans le helper : remplacez l'URL de base par celle de votre endpoint Beeceptor.
3. Exécutez l'application console : Testez l'envoi et la réception de données.

4. Allez plus loin :

- Observez comment la classe HttpClient gère les verbes HTTP.
- **Ajouter une modification partielle avec PATCH**
- Améliorez le code pour ajouter une gestion d'erreurs (try/catch) ou pour formater l'affichage des résultats.
- Amusez-vous à modifier les objets envoyés pour voir comment l'API réagit à de nouveaux champs.

À REMETTRE DANS L'ÉA

1. L'URL de votre endpoint Beeceptor.
2. Une capture d'écran de votre règle « Mocking Rule ».
3. Une capture d'écran d'une réponse de requête GET réussie dans Postman.
4. Une capture d'écran de votre application console confirmant le **succès d'une opération POST, d'une opération PUT et d'une opération PATCH**.

BON TRAVAIL !