Compte-rendu de projet

Framework Symfony et React

Calculateur de temps de plongée

Projet réalisé par

Julien Marquet

Titouan Allain

Supervisé par

Christophe Vignaud

Fabien Canu

Semestre 1

Année 2020 - 2021

# Index

1. Symfony
   1. Entité
   2. API
      1. URL des API
      2. Description des API
         1. api\_profondeur
         2. api\_profondeur\_by\_depth
         3. api\_temps
         4. api\_temps\_by\_depth\_and\_time
         5. api\_defaultParam
   3. Méthodes CRUD
      1. Liste des entités CRUD-able
      2. Exemples génériques des méthodes CRUD
         1. Créer
            1. readAll()
            2. read($id)
            3. readSelector()
         2. Lire
            1. New(Request $request)
         3. Éditer
            1. checkEditAvailable($id)
            2. edit(Request $request, Entité $entite)
            3. editSelector()
         4. Supprimer
            1. delete($id)
            2. deleteSelector()
2. React
3. Ressources

# Symfony

## Entité

Le cahier de charge initiale stipulait la nécessité de 3 entités pour réaliser la fonction première de l’application du projet, le calcul du temps de plongée en fonction d’une table de plongée (TablePlongee), d’une profondeur (Profondeur) et d’un temps donné (Temps).

Pour faciliter l’utilisateur dans l’interface de calcul de temps de plongée en front-end, nous avons rajouter une entité de paramètre par défaut (DefaultParam) pour permettre au service front-end de donner à l’utilisateur une configuration par défaut.

Diagram

Description automatically generated

## API

La mise en place de ces API permettent un partage de ressources de base de données en backend pour notamment le serveur front-end.

### URL des API

*Attention : les liens cliquables ne fonctionneront uniquement si vous avez le serveur Symfony en cours d’exécution sur votre environnement de développement en local.*

<http://localhost:8000/api/profondeur>

[http://localhost:8000/api/profondeur/depth/{depth}](http://localhost:8000/api/profondeur/depth/%7bdepth%7d)

[http://localhost:8000/api/temps](http://localhost:8000/api/tempsr)

<http://localhost:8000/api/defaultParam> [http://localhost:8000/api/temps/depth/{depth}/time/{time}](http://localhost:8000/api/temps/depth/%7bdepth%7d/time/%7btime%7d)

### Description des API

* @Route("/profondeur", name="api\_profondeur")

API pour avoir toutes les entrées de l’entité Profondeur.

* @Route("/profondeur/depth/{depth}", name="api\_profondeur\_by\_depth")

API pour avoir la première entrée de l’entité Profondeur qui à la bonne profondeur choisie.

* @Route("/temps", name="api\_temps")

API pour avoir toutes les entrées de l’entité Temps.

* @Route("/temps/depth/{depth}/time/{time}", name="api\_Temps\_by\_Depth\_and\_Time")

API pour avoir la première entrée de Temps en fonction d’une valeur de temps et d’une valeur de profondeur.

* @Route("/defaultParam", name="api\_defaultParam")

API pour avoir toutes les entrées de l’entité DefaultParam.

## Méthodes CRUD

### Liste des entités CRUD-able

Les entités ayant les fonctionnalités CRUD :

* DefaultParam
* TablePlongee
* Profondeur
* Temps

### Exemples génériques des méthodes CRUD

#### Créer

##### new(Request $request)

Le paramètre $request, provennant du composant HttpFoundation, permet de faire une requête sur l’entité afin de récupérer son formulaire.

Fonction créant un objet de la classe de l’entité choisie pour l’ajouter en base de données, avec pour valeur de champs ceux provenant du formulaire de cette même entité.

Si le formulaire est bien valide, le manageur d’entité de Doctrine nous insère cet objet selon le formulaire en base de données.

#### Lire

##### readAll()

Fonction demandant la récupération de toutes le entrée de l’entité choisie, grâce à Doctrine et la fonction de findAll() du Repository de l’entité. Les données récupérées sont ensuite affichées comme paramètre du twig readAll.html.twig.

##### read($id)

Le paramètre $id est un entier qui permet de definir l’ID de l’entrée à lire.

Fonction demandant la récupération de l’entré d’ID défini par $id de l’entité choisie. La donnée récupérée est ensuite afficher comme paramètre du twig read.html.twig. Dans le cas où l’entrée demandée n’existe pas dans la base, un message d’erreur apparait.

##### readSelector()

Fonction renvoyant sur le twig selector.html.twig avec comme parametre de twig crud\_method égale à « read » et entity égale au nom de l’entité choisie.

#### Editer