Convention de codage pour le projet « Keymada »

1. **Convention d’écriture**

Afin que tout le monde puisse comprendre l'intérêt d'une telle variable, fonction à tel endroit du code, il est judicieux de bien choisir son nom et de bien commenter le code.

1. Langue

Veillez à ne coder qu'en une seule langue. C'est-à-dire qu'il faut éviter de mélanger français et anglais

Tous les noms des fichiers, les variables, les constantes, les fonctions, les classes ainsi que les commentaires dans le code sont écrit en français

1. Variables

Donner des noms simples et descriptifs

Première lettre du premier mot en minuscule et la première lettre du reste des mots en majuscule

N'utiliser que des lettres [a-z][A-Z] et [0-9]

Ex : monVariable, nomUtilisateur

1. Constantes

Donner des noms simples et descriptifs

Tout en majuscule

Séparer les mots par des underscore

N'utiliser que des lettres [a-z][A-Z] et [0-9]

Ex : MA\_CONSTANTE

1. Fonctions

Donner des noms simples et descriptifs

Première lettre du premier mot en minuscule et la première lettre du reste des mots en majuscule

N'utiliser que des lettres [a-z][A-Z] et [0-9]

Ex : maFonction, sauvegardeEvenement

1. Classes

Donner des noms simples et descriptifs

Première lettre de chaque des mots en majuscule

N'utiliser que des lettres [a-z][A-Z] et [0-9]

Ex : maClasse, SelecteurFormation

1. Commentaires

Ecrire un simple commentaire utile et le mettre au début de bloc

1. Indentation

L'ensemble du code devrait être indenté de façon similaire

Exemple

**public** class MyApp

{

**public** **static** void main (String[] args)

{

System.out.println("Hello World");

}

}

1. Les accolades

On place les accolades après le nom de fonction

Exemple :

[function](http://www.php.net/manual-lookup.php?pattern=function&lang=fr&scope=quickref) checkpoints($x, $y) {

...

}

1. Les espaces

Veillez à toujours bien espacer les variables et les opérateurs.

Exemple : $total = $apple + $pear;

1. Base de données, Collections, champs

**Pour la base de données** : nom en minuscules, pour éviter les problèmes de sensibilité aux cas, suivi du « undercsore »et le mot « db »

Exemple : keymada\_db

**Pour la collection** : nom en minuscules, et en plurielles

Exemple : utilisateurs, formations

**Pour les champs** : première lettre du premier mot en minuscule et la première lettre du reste des mots en majuscule

Exemple :anneNaissance

1. **Nom pour les entités**

|  |  |
| --- | --- |
|  | **Nom dans le code et dans la base de données** |
| Utilisateur ou candidat | Utilisateurs |
| Entreprise/Etablissement | Institutions |
| Formation | Formations |
| Evènement | evenements |
| Emploi | Emplois |

1. **Nom fichier**

Le nom de fichier qui sera saisi dans la ligne de commande doit être en type CamelCase : en majuscule la première lettre de chaque mot

1. **Convention API REST**

**Format de l’URL** :

-URI RESTful doit se référer à une ressource qui est une chose (nom) au lieu de se référer à une action (verbe), et le nom de ressources est en pluriel

http://url\_base/applications/,

* où url\_base est le nom d'hôte et le chemin vers le serveur ;
* applications est le donnée qui s’appelle ressources avec une forme plurielle

Exemple : <http://localhost:1337/utilisateurs>

On utilise des verbes pour exprimer les actions ou opérations

#### Gestion d’erreur : renvoyer des codes HTTP appropriés:

* SUCCES

200 ok**:** Code de réussite, utilisé spécialement pour les premières requêtes GET réussies ou le contenu mis à jour PUT / PATCH

201 ok : Indique qu'une ressource a été créée. Répondant généralement aux demandes PUT et POST

204 OK : La demande a réussi mais il n'y a rien à montrer, la ressource a été supprimée avec succès ; utilisé après une DELETE réussie

* CLIENT ERROR

400 – Bad request : Erreur générale pour une demande qui ne peut pas être traitée  
401 - unauthorized : La requête nécessite une authentification utilisateur,   
403 - Forbidden : Le serveur a compris la demande, mais la refuser parce que l'accès n'est pas autorisé.   
404 – Not found - Il n'y a aucune ressource derrière l'URI, la ressource demandée n’existe pas

* SERVER ERROR

500 – Internal Server Error : un problème s’est produit sur le serveur

**Requête HTTP appropriés**

| **Verbe HTTP** |  |  |  |  |  |  |  |  | **Correspondance CRUD** |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| GET |  |  |  |  |  |  |  |  | READ |  |  |
| POST |  |  |  |  |  |  |  |  | CREATE |  |  |
| PUT |  |  |  |  |  |  |  |  | UPDATE/CREATE |  |  |
| PATCH |  |  |  |  |  |  |  |  | UPDATE |  |  |
| DELETE |  |  |  |  |  |  |  |  | DELETE |  |  |

**Filtre et tri sur les ressources**

**Exemple :**

http:// localhost:1337/utilisateurs?fitlre=policier&tri=asc

Références utilisés :

<http://www.learnit.net.in/2016/03/schema-design-and-naming-conventions-in.html>

<http://arkusnexus.com/2016/09/12/coding-guidelines-mongodb/>

<https://restfulapi.net/resource-naming/>

<http://www.vinaysahni.com/best-practices-for-a-pragmatic-restful-api#restful>