Inscription à un cours

Laetitia Guidetti  
Chef de projet : Cindy Hardegger  
ETML mai-juin 2021

Table des matières

[1 Analyse préliminaire 3](#_Toc72243776)

[1.1 Introduction 3](#_Toc72243777)

[1.2 Objectifs 3](#_Toc72243778)

[1.3 Méthodologie de gestion de projet 3](#_Toc72243779)

[2 Analyse et Conception 4](#_Toc72243780)

[2.1 Planification initial 4](#_Toc72243781)

[2.2 Symfony 4](#_Toc72243782)

[2.3 Bootstrap 4](#_Toc72243783)

[2.4 Mockup 5](#_Toc72243784)

[2.5 MCD et MLD 6](#_Toc72243785)

[2.6 Doctrine ORM 7](#_Toc72243786)

[3 Réalisation 9](#_Toc72243787)

[3.1 Routes 9](#_Toc72243788)

[3.2 Entités 9](#_Toc72243789)

[3.2.1 Environnement 9](#_Toc72243790)

[3.2.2 Création d’une entité 10](#_Toc72243791)

[3.2.3 Migration 13](#_Toc72243792)

[4 Tests 14](#_Toc72243793)

[5 Conclusion 14](#_Toc72243794)

[6 Glossaire 14](#_Toc72243795)

[7 Sources et références 15](#_Toc72243796)

[8 Annexes 15](#_Toc72243797)

[8.1 Journal de travail 15](#_Toc72243798)

# Analyse préliminaire

## Introduction

## Objectifs

## Méthodologie de gestion de projet

# Analyse et Conception

## Planification initial

## Symfony

Dans ce projet utiliser un framework PHP n’est pas imposé. Cependant, je souhaite utiliser Symfony et une explication s’impose.

Lors de la création d’un site Web certaines fonctionnalités reviennent très souvent : La gestion de l’authentification des utilisateurs, un formulaire de contact ou encore la validation des données. Il s’agit de tâches redondantes, ce serait une perte de temps de les recréer à chaque nouveau site. Un framework PHP permet alors de fournir différent composants permettant d’implémenter directement ses fonctionnalités sans avoir besoin de les recréer. Ses composants ont chacun déjà été utilisé par des milliers des sites, par conséquent un grand nombre de bugs et de faille de sécurités ont déjà été trouvés et corrigés.

Un framework permet une fois un minimum maitriser de gagner beaucoup de temps tout en améliorant la sécurité globale du site. Le défaut principal est qu’au début, il est assez dur d’apprendre un framework PHP, car il nécessite de comprendre son fonctionnement.

Parmi tous les framework PHP, le choix s’est porté sur Symfony. Il s’agit de l’un des plus utilisé à travers le monde et il est le premier dont j’ai entendu parlé. Il intègre également l’architecture MVC nécessaire dans ce projet.

## Bootstrap

## Mockup

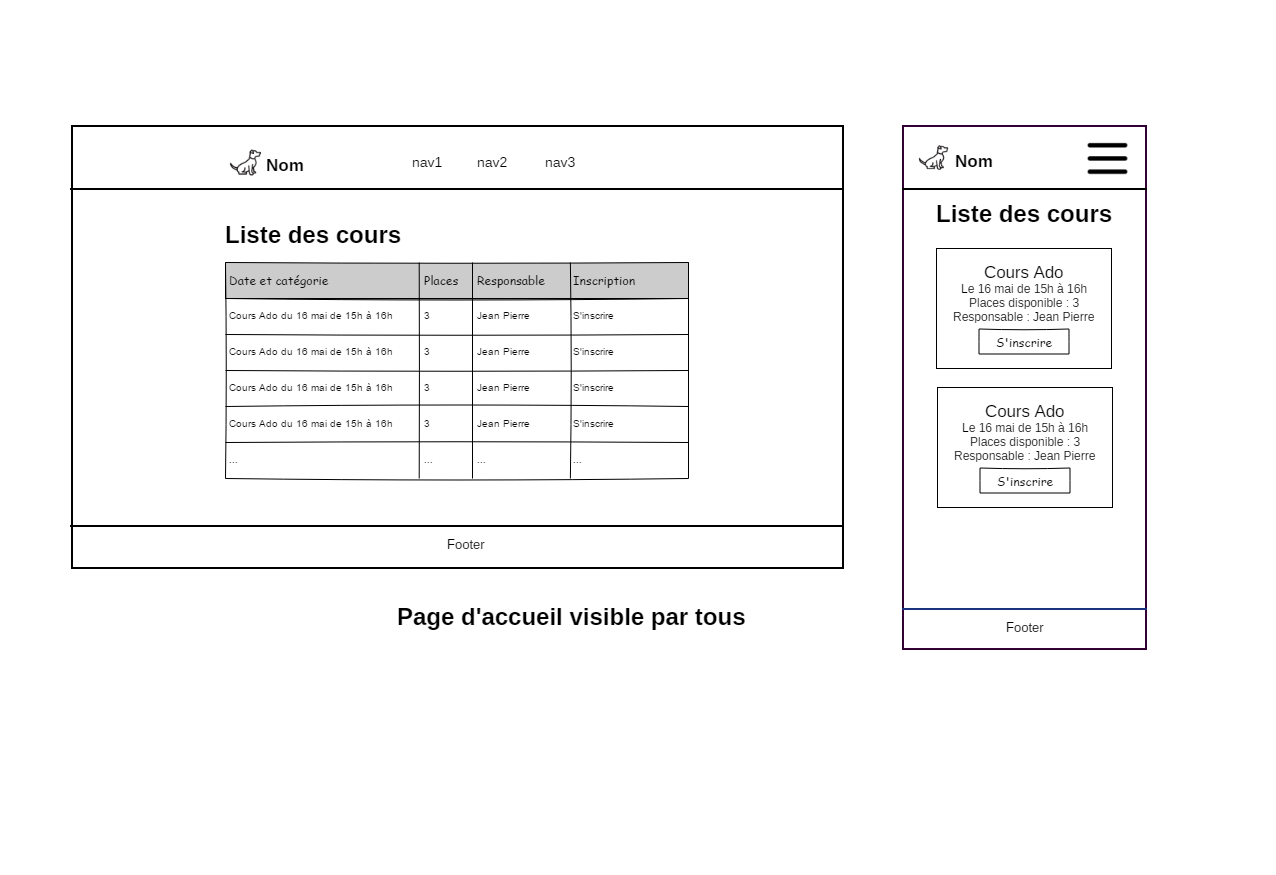
Une mockup est une maquette permettant de représenter le site web de manière simpliste.

Figure 1 Mockup de la page d'accueil

La page d’accueil du site sert à afficher les différents cours et permet aux utilisateurs de s’y inscrire. Par conséquent, l’aspect graphique doit rester assez simple afin de ne pas rendre la tâche difficile aux utilisateurs. L’affichage sur téléphone est modifié car la largeur de ces appareils ne permet pas d’afficher les grands tableaux.

Le formulaire d’ajout d’un cours ne contient que les informations utiles pour l’administrateur tout en restant intuitif. Ce mockup sert de base à toutes les pages du site utilisant des formulaires comme pour l’inscription d’un utilisateur à un cours.

Figure 2 Mockup du formulaire d'ajout de cours

## MCD et MLD

La modélisation d’une base de données ce faite en 3 étapes.

1. Le modèle conceptuel des données (MCD)
2. Le modèle logique de données (MLD)
3. Le modèle physique de données (MPD)

Pour réaliser une base de données plusieurs règles sont à respecter.

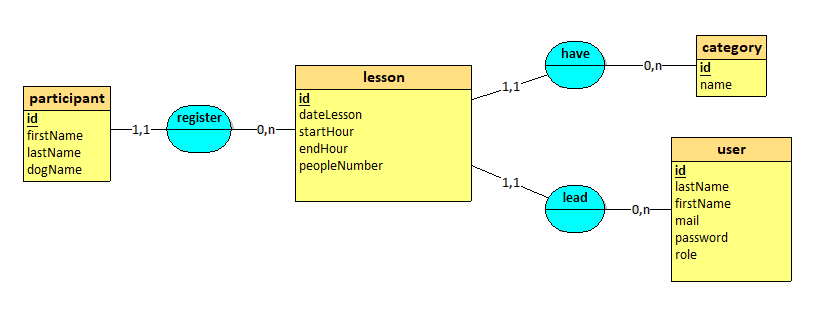
* Non redondance : chaque information n’apparaît qu’une fois
* Atomicité : une seule information par champ
* Unicité : chaque champ est identifiable de manière unique

Figure 3 MCD de la base de données

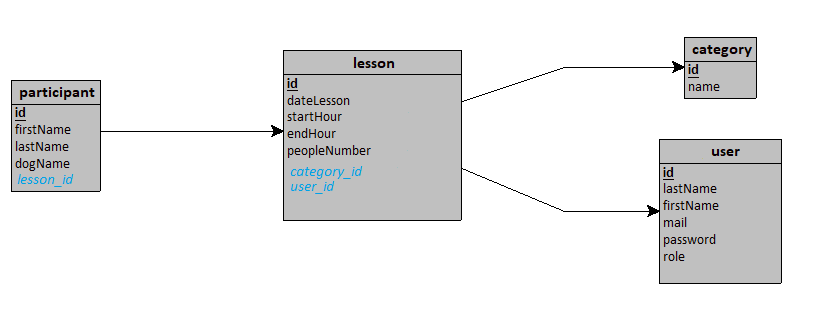
Dans le CDC, une liste des caractéristiques d’un cours est donnée. Cependant, créer un type d’entité directement d’après cette liste ne respecterait pas les règles d’élaboration d’une BD. L’une des caractéristiques est qu’un cours possède « Une catégorie de cours (ex : Ado) ». Ajouter, sur le type d’entité « lesson », l’attribut « category » serait contraire au principe de la non redondance. En effet, plusieurs cours pourraient avoir comme catégorie « Ado », il y aurait alors une redondance s’il s’agissait d’un attribut de « lesson ». Un nouveau type d’entité est donc utilisé afin de respecter cette règle. On remarque également que pour le nom des participants, 2 attributs sont prévus l’un pour le nom de famille et l’autre pour le prénom, afin de respecter la règle d’atomicité.

Figure 4 MLD de la base de données

Le MLD est basé sur le MCD et représente les futurs tables et colonnes de la base de données.

## Doctrine ORM

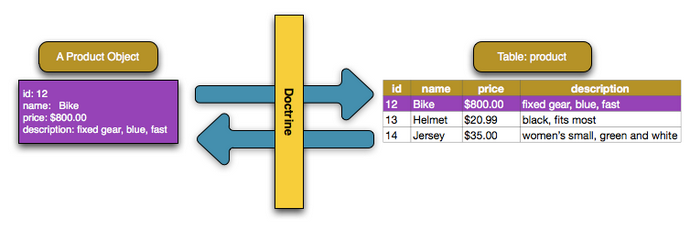
Doctrine est l’ORM (object-relational mapping) utilisé par Symfony. Un ORM est un couche d’abstraction qui synchronise les objets PHP avec une base de données relationnelle. Son but est de créer un ensemble de classe permettant de manipuler les tables et données de la BD. Il permet de transformer les objets PHP en écriture SQL ce qui permet notamment de générer des requêtes SQL.

Figure 5 Fonctionnement de Doctrine, source : https://symfony.com/doc/current/doctrine.html

C’est donc via Doctrine que la base de donnée va être créée ainsi que ses différentes tables en se basant sur le MLD. Il va permettre de créer des classes appelées entités. A partir de ses dernières, une base de données correspondante pourra être générée automatiquement.

# Réalisation

## Routes

Quand un utilisateur clique sur un lien ou écrit une URL, la page correspondante doit s’afficher. Le routage permet de définir quelle action doit être effectuer en fonction de la requête reçue par l’application Web. L’action renverra ensuite les différentes informations à afficher.

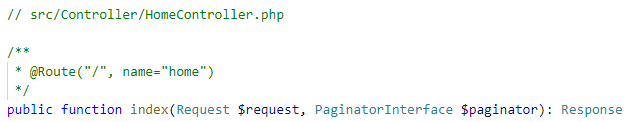
Il est possible de définir les routes de plusieurs façons, : via des attributs ou des annotations, dans des fichiers YAML, XML ou PHP. Selon les [bonnes pratiques](https://symfony.com/doc/current/best_practices.html#use-attributes-or-annotations-to-configure-routing-caching-and-security)[[1]](#footnote-1) de Symfony, il est recommandé de les définir via des attributs ou des annotations.

Figure 6 Route définie par une annotation

Dans ce cas, quand l’utilisateur va sur l’adresse « / », la méthode index est appelée. Elle se charge de répondre à la demande en affichant la page correspondante.

## Entités

### Environnement

Pour créer et utiliser des entités, il faut que l’environnement soit correctement configuré. Cela demande une base de données vide et les informations pour s’y connecter. Elles doivent être définie dans le fichier .env à la racine du projet sous la variable d’environnement « DATABASE\_URL ». Cette ligne contient en premier le nom et le mot de passe de l’utilisateur qui accède à la BD. Puis l’adresse de l’hôte et le nom de la base de données sont renseignés.

Figure 7 Information de la base de données dans le fichier .env

C:\Users\laeguidetti\Desktop\Screen\Database\CreateDB.PNGUne base de données vide correspondant à ses informations peut être alors généré via une commande. Une erreur sera soulevée si une BD portant son nom existe déjà.

Figure 8 Commande créant la base de données

Il est maintenant possible de créer les entités qui permettront de générer les tables dans la base de données.

### Création d’une entité

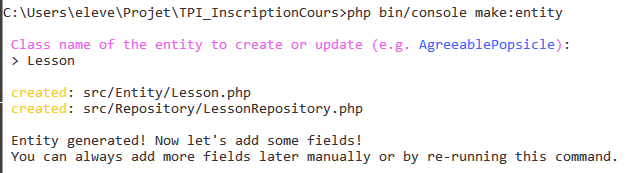
La création d’une entité se fait via une ligne de commande.

Figure 9 Création d'une entité

Après avoir donné le nom de l’entité, 2 fichiers sont automatiquement créés. Le premier « Lesson.php » contient la classe « Lesson ». Le deuxième est le fichier « LessonRepository.php », il permet d’exécuter des requêtes de base et de créer des requêtes précises liées.

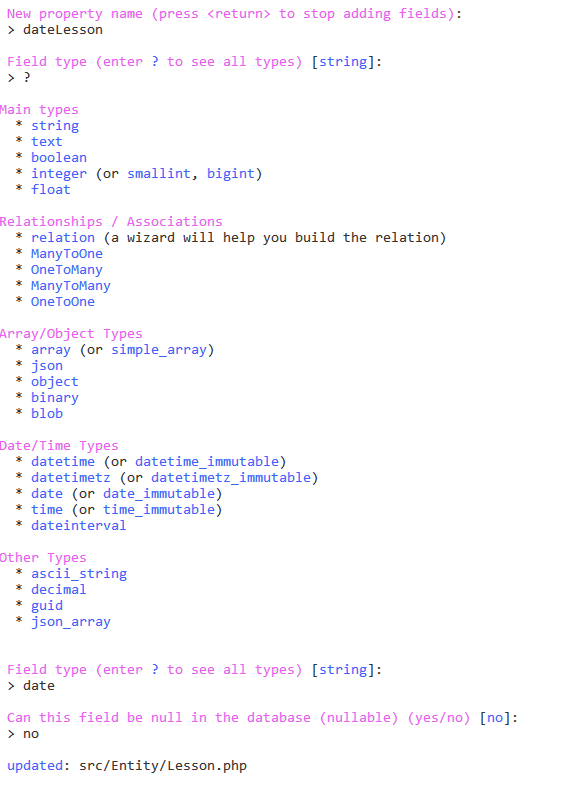
L’invite de commande proposera ensuite d’ajouter des propriétés à l’entités créées.

Figure 10 Ajouter une propriété à une entité

Plusieurs questions seront alors posées pour configurer la propriété correctement. Le fichier « Lesson.php » est alors mis à jour en conséquence. Il est possible dans ce dernier de configurer de manière plus précisée les propriétés comme ajouter une valeur par défaut. Ce fichier génère également des getter et setter pour chaque propriété afin de faciliter leur utilisation.

Après avoir créé toutes les entités, il manque les relations entre elles.

Figure 11 Création d'une relation

Pour modifier une entité déjà créée, il suffit lors d’un « make:entity » d’indiquer son nom. La console reconnait automatiquement que l’entité existe déjà et permet de lui ajouter de nouvelles propriétés. Comme son nom l’indique le type « relation » permet de définir les relations. Il suffit ensuite de préciser avec quel entité la relation se fait et son type. Cela mettra à jour les 2 entités afin que la relation soit valide.

Toutes ses opérations sont également réalisables en modifiant à la main les divers fichier d’entités.

### Migration

Toutes les entités et leurs relations sont créées, cependant la base de données ne contient toujours aucunes tables. Il faut alors migrer toutes les informations présentent dans les classes entités dans cette dernière. Une commande permet de générer automatiquement un fichier contenant des requêtes SQL correspondant aux entités.

Figure 12 Migration

Une deuxième commande permet d’exécuter tous les fichiers de migrations qui n’ont pas déjà été opéré. La base de données est alors modifiée en conséquence. Si plus tard des modifications sur une entité surviennent, il suffit de recréer un fichier de migration et de l’exécuter.

Il existe d’autres commandes liées aux migrations. Il est possible de supprimer la dernière migration, d’obtenir la liste de toutes les migrations ou encore la version actuelle. Ses cas spécifiques ne sont pas abordé ici, mais ils sont présent dans la [documentation](https://symfony.com/doc/current/bundles/DoctrineMigrationsBundle/index.html)[[2]](#footnote-2) de Symfony.

# Tests

# Conclusion

# Glossaire

Alphabétiquement trié

# Sources et références

|  |  |
| --- | --- |
| Information | Lien |
| Mockup | https://www.journaldunet.fr/web-tech/dictionnaire-du-webmastering/1203293-mockup-definition-traduction/ |
| Doctrine ORM | https://symfony.com/doc/current/doctrine.html |
| Information sur Symfony | https://symfony.com/why-use-a-framework |
| Bonnes pratiques Symfony | https://symfony.com/doc/current/best\_practices.html |
| Migration Symfony | https://symfony.com/doc/current/bundles/DoctrineMigrationsBundle/index.html |

# Annexes

## Journal de travail

1. https://symfony.com/doc/current/best\_practices.html#use-attributes-or-annotations-to-configure-routing-caching-and-security [↑](#footnote-ref-1)
2. https://symfony.com/doc/current/bundles/DoctrineMigrationsBundle/index.html [↑](#footnote-ref-2)