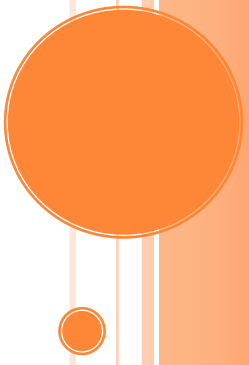
SPECIFICATION INTERNE

MOOC'Line

Plateforme d'apprentissage et auto-évaluation



Spécification interne

Plateforme d'apprentissage et auto-évaluation

Table des matières

Introduction

1.1. But

Ce document décrit l'architecture de l'application web « ***Mooc'line*** » qui est une plateforme de cours en ligne.

1.2. Portée

Ce design concerne la version qui sera remise à la fin de la deuxième itération. Il représente aussi une base de celui de la version finale qui sera remise à la soutenance finale, même s'il évolue entre les différentes itérations. Cette architecture est conçue de façon à faciliter la réalisation de versions améliorées dans le futur en se basant sur l'architecture fourni par le framework Symfony2.

L'Outil Symfony2

Pour la réalisation de notre application web, nous utilisons le framework Symfony2. « C'est un framework MVC libre écrit en PHP 5 ». Il offre un cadre de travail qui facile et accélère le développement des applications telles que la nôtre.

2.1. Architecture des fichiers

Symfony2 est organisé autour de quatre répertoires principaux :

- **/app :** Ce répertoire contient tout ce qui concerne le noyau de Symfony2. Il ne contient pas les codes sources de notre application web. En fait, c'est simplement pour séparer le code source, qui fait la logique de l'application web, du reste. Et ce reste c'est : la configuration, le cache, les fichiers logs, etc. Ce sont des fichiers qui concernent l'entièreté de notre application, contrairement aux fichiers de code source qui seront découpés par fonctionnalité.

- **/src :** C'est le répertoire dans lequel on mettra le code source! Dans ce répertoire, nous organiserons notre code en bundles, des briques de notre application, dont nous verrons la définition plus loin.

- **/vendor :** Ce répertoire contient toutes les bibliothèques externes à notre application. Dans ces bibliothèques externes, y compris Symfony2 lui-même! C'est dans ce répertoire qu'on trouve des bibliothèques comme Doctrine (pour la gestion de la base de données), Twig (moteur de Template pour les vues), SwiftMailer (pour l'envoi des mails), etc. Ces bibliothèques sont détaillées plus loin.

- **/web :** Ce répertoire contient tous les fichiers destinés aux visiteurs : images, fichiers CSS et JavaScript, etc. Il contient également le contrôleur frontal (c'est le point d'entrée de notre application). En fait, c'est le seul répertoire qui est accessible aux visiteurs. Les autres répertoires ne le sont pas.

2.2. Environnements de travail

En fait, il y a deux points d'entrée pour notre application. L'objectif est de répondre au mieux suivant la personne qui visite le site :

- Un développeur a besoin d'informations sur la page afin de l'aider à développer. En cas d'erreur, il veut tous les détails pour pouvoir déboguer facilement. Il n'a pas besoin de rapidité.
- Un visiteur normal n'a pas besoin d'informations particulières sur la page. En cas d'erreur, l'origine de celle-ci ne l'intéresse pas du tout, il veut juste retourner d'où il vient. Par contre, il veut que le site soit le plus rapide possible à charger.

C'est pourquoi Symfony2 nous offre plusieurs environnements de travail :

- L'environnement de développement, appelé « dev », accessible en utilisant le contrôleur frontal app_dev.php. C'est l'environnement que l'on utilise toujours pour développer.
- L'environnement de production, appelé « prod », accessible en utilisant le contrôleur frontal app.php.

2.3. Architecture conceptuelle : le MVC

Symfony2 utilise le fameux « design pattern » MVC qui très répandu dans la conception et développement des applications web. Le MVC signifie « Modèle, Vue, Contrôleur ». Ce pattern permet de faire un découpage en couches selon leurs logiques :

- Le Contrôleur (ou Controller) : son rôle est de générer la réponse à la requête HTTP demandée par notre visiteur. Il est la couche qui se charge d'analyser et de traiter la requête de l'utilisateur. Le contrôleur contient la logique de notre site internet et va

se contenter « d'utiliser » les autres composants : les modèles et les vues. Concrètement, un contrôleur va récupérer, par exemple, les informations sur l'utilisateur courant, vérifier qu'il a le droit d'accéder à la page de cours si c'est un étudiant, récupérer la liste de cours auxquels il est inscrit ou demander la page du formulaire d'édition de cours si c'est un enseignant.

- Le Modèle (ou Model) : son rôle est de gérer les données et le contenu. Si nous reprenons l'exemple précédent, quand le contrôleur « récupère » les informations de l'utilisateur courant, il va en fait faire appel au modèle. C'est le modèle qui sait comment récupérer les informations via une requête SQL à la base de données, mais ce peut être depuis un fichier xml pour la partie cours. Cela permet au contrôleur de manipuler les données, mais sans savoir comment elles sont stockées, gérées. C'est une couche d'abstraction.

- La Vue (ou View) : son rôle est d'afficher les pages. Pour afficher les formulaires, la liste des cours, etc… le contrôleur ne fait qu'appeler la bonne vue et afficher cette vue sans savoir vraiment ce qu'il y a dedans.

Décomposition par modules (briques)

Symfony2 permet de découper une application web en différentes briques indépendantes mais qui interagissent entre elles. Nous avons utilisés ce concept pour subdiviser notre système en trois briques qui correspondent aux trois principaux modules de notre application :

3.1. Gestion compte :

Ce bundle gère toute la partie qui concerne l'inscription à la plateforme (étudiants, organismes), l'envoi de mail de vérification, l'authentification), réinitialisation de mot de passe en cas de problème d'authentification et tout ce qui touche à la gestion des profils : édition des informations personnelles (identité, avatar, etc..).

Ci-dessous le diagramme de séquence montrant un scénario d'inscription :

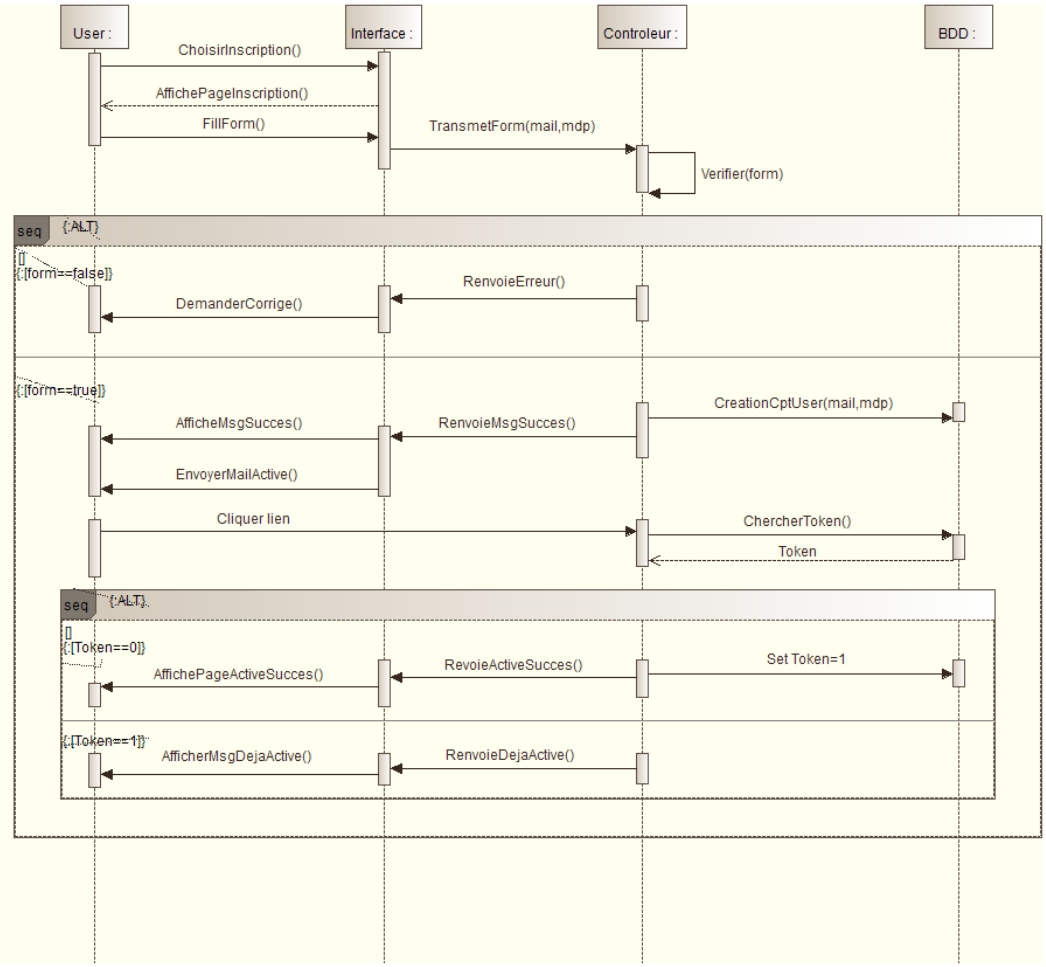


Fig. 1 Diagramme de séquence : Scénario d'inscription.

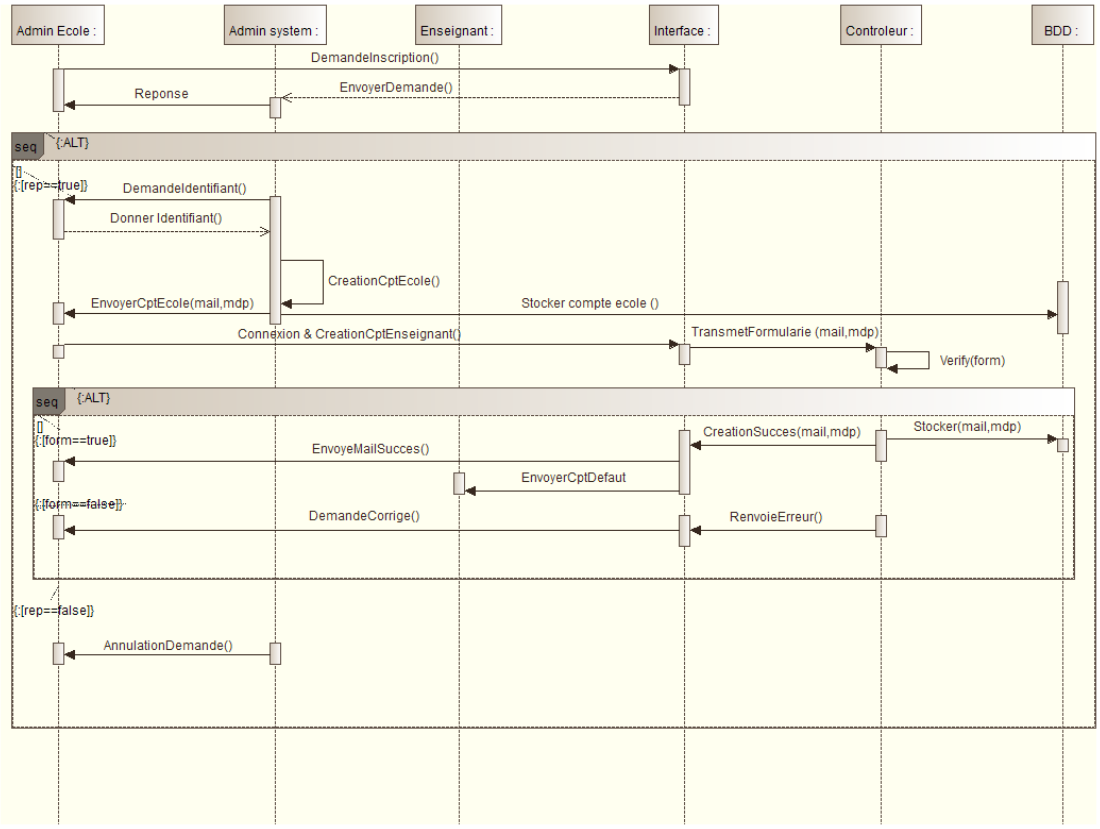


Fig. 2 scénario création de compte d'une école et ses enseignants

3.2. Gestion des Cours :

C'est dans ce bundle que nous trouvons tout ce qui concerne la création, la modification, la suppression de cours, d'exercice et la correction automatique des exercices.

Ci-dessous le diagramme de séquence montrant un scénario de création de cours et d'exercice par un enseignant.

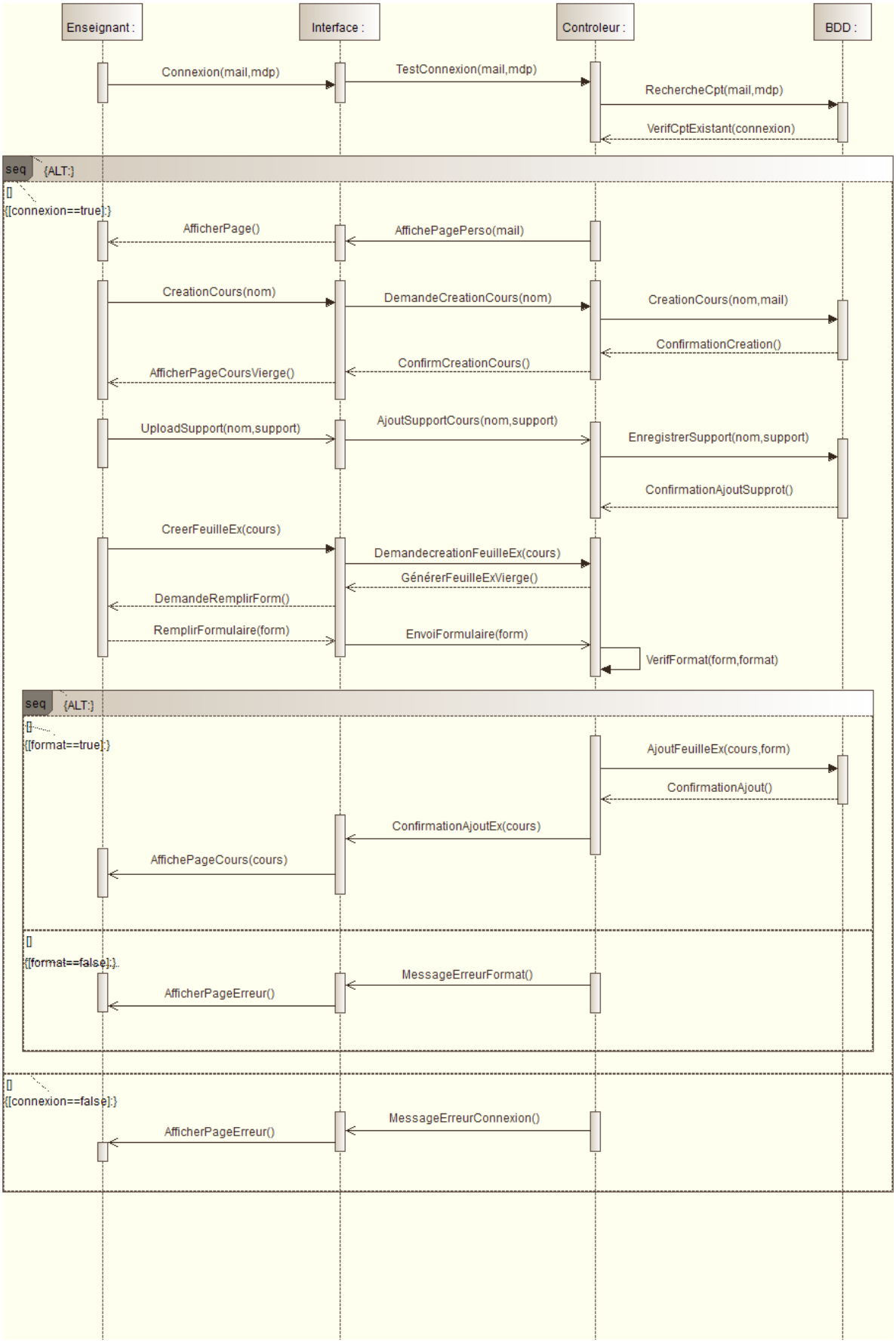


Fig. 3 Scénario de création de cours et d'exercice.

3.3. Forum :

Notre application permet aux utilisateurs d'échanger à travers un forum où ils pourront poster, consulter des questions. Le forum est accessible par tous mais seuls ceux qui sont inscrits, sont autorisés à poster des messages. Cette brique de l'application met en œuvre ce forum.

Ci-dessous le diagramme de séquence montrant un scénario où un étudiant poste une question sur le forum de la plateforme.

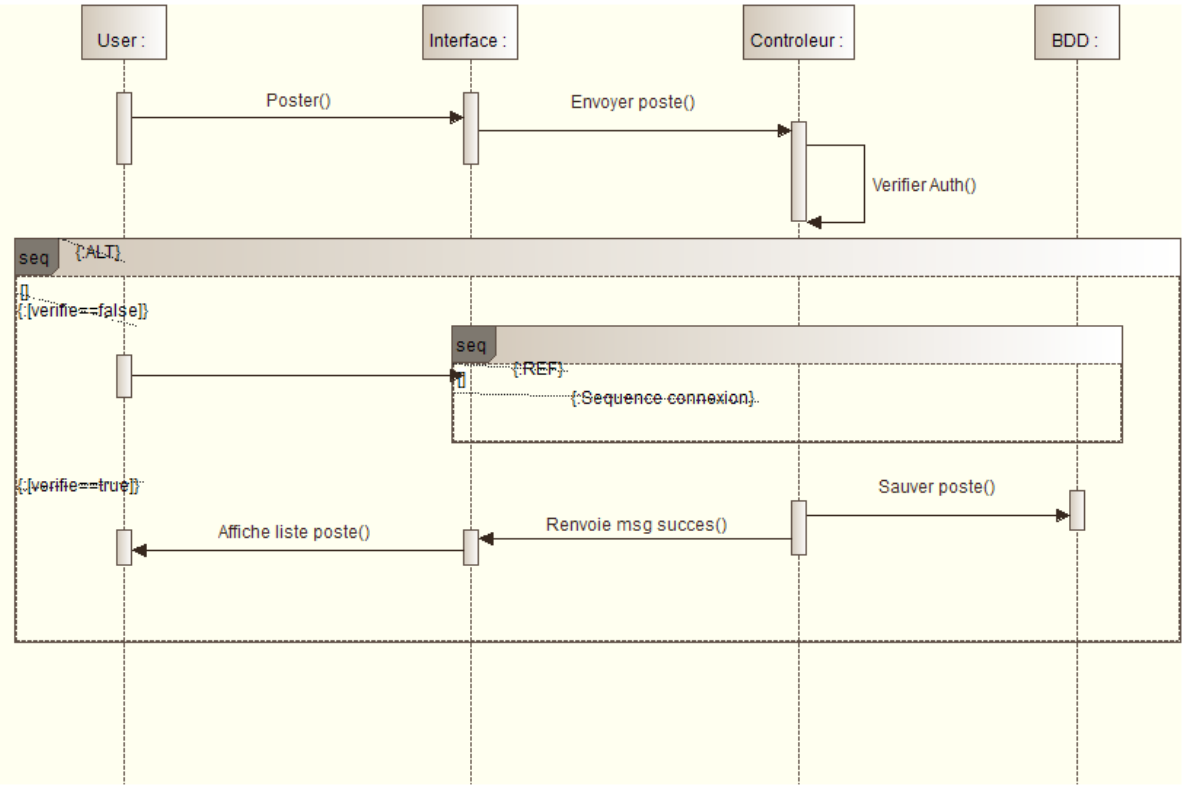


Fig. 4 post d'un étudiant sur le forum de la plateforme