

# Projet web - UnsapaIPW

Léo Unbekandt - Guillaume Paran - Lucas Saurel

Mai 2012

# Table des matières

|   |   |
|---|---|
| Introduction . . . . .  | 2 |
| 0.1 Fonctionnalités de l'application . . . . .                  | 2 |
| 0.1.1 Gestion des rôles . . . . .                               | 2 |
| 0.1.2 Les différentes <i>user stories</i> . . . . .             | 2 |
| 0.1.3 Ce que nous n'avons pas fait . . . . .                    | 5 |
| 0.2 Organisation de l'équipe . . . . .                          | 5 |
| 0.3 Schéma de données . . . . .                                 | 5 |
| 0.3.1 User . . . . .  | 5 |
| 0.3.2 Promo . . . . .   | 6 |
| 0.3.3 Exam . . . . .  | 7 |
| 0.3.4 Record . . . . .  | 7 |
| 0.4 Réalisation Technique . . . . .                             | 7 |
| 0.4.1 UserBundle . . . . .                                      | 7 |
| 0.4.2 Les tests avec PHPUnit . . . . .                          | 8 |
| 0.4.3 La documentation développeur avec phpDocumentor . . . . . | 8 |
| Conclusion . . . . .  | 8 |

# Introduction

## 0.1 Fonctionnalités de l'application

### 0.1.1 Gestion des rôles

L'application web que nous avons réalisée est un système de gestion d'examens. On distingue trois catégories d'utilisateurs :

- Les étudiants : ils consultent les examens auxquels leur promotion est inscrite et déposent leurs compositions.
- Les responsables d'examens : ils créent des examens et notent les étudiants.
- L'administrateur : il gère l'ensemble des utilisateurs, des promotions et des examens.

### 0.1.2 Les différentes *user stories*

**L'utilisateur consulte la liste des examens auxquels il est inscrit**

```
Route : /exams
Controller : ExamsController
Action : indexAction
```

**L'utilisateur s'authentifie**

Il peut soit cliquer manuellement sur *se connecter* ou il lui sera demandé de s'authentifier lorsqu'il tentera d'accéder à des pages où un utilisateur inconnu n'a pas les droits. Les mots de passes sont stockés en sha1 dans la base de données, personne à part l'utilisateur ne peut le récupérer.

```
Route : /login
Controller : FOSUserBundle:SecurityController
```

**L'utilisateur s'inscrit**

Lorsqu'un utilisateur s'inscrit son compte est immédiatement activé, mais nous pouvons configurer notre application qu'un mail soit envoyé avec une URL de validation. (

Chemin : app/config/config.yml

```
Route : /register
Controller : FOSUserBundle:RegistrationController
```

### **L'utilisateur dépose un examen**

Parmi les examens auxquels il est inscrit un étudiant peut proposer un rendu pour l'un d'entre eux. Les fichiers DOC, DOCX, PDF et ZIP sont acceptés. Nous faisons la vérification par rapport aux mime-types et non seulement les extensions.

Lorsqu'il choisit l'examen pour lequel rendre un document, il sera averti s'il a déjà rendu quelque chose. Et l'interface lui proposera de télécharger l'ancienne version de son travail.

Page de soumission :

```
Route : /register/submit
Controller : ExamsController
Action : submitAction
```

Téléchargement de fichier :

```
Route : /download/{:examid}/{:userid}
Controller : AttendController
Action : downloadAction
```

### **L'utilisateur crée un examen**

Si un utilisateur a le rôle de responsable de TD, il a la possibilité d'ajouter de créer des examens.

```
Route : /exams/add
Controller : ExamsController
Action : addAction
```

### **L'utilisateur associe des étudiants à un examen**

Pour cela il y a 3 possibilités :

- Lors de l'ajout d'un examen, il choisit une promotion. Par défaut tous les étudiants de cette promotion seront concernés par cet examen.
- Lors de l'ajout d'un exam, après avoir choisi une promotion, il peut cliquer sur détails, et cela lui permettra de voir tous les étudiants de la promotion sélectionnée, et il pourra choisir, étudiant par étudiant qui participe à l'examen.
- Sur la page listant les examens dont il est responsable, il peut cliquer sur "Gérer les étudiants", et il accèdera à une page où comme dans le point ci-dessus, il pourra gérer plus finement les étudiants qui sont concernés :

```
Route : /exams/{:examid}/students
Controller : AttendController
Action : examChoiceAction
```

### **Associer une note à un document déposé**

Lorsqu'un examen est terminé, le responsable peut commencer à noter les rendus. Il ne peut pas le faire avant, car nous laissons aux étudiants la possibilité de modifier leur rendu jusqu'au dernier jour. Et noter un rendu qui n'est pas définitif ne sert à rien.

Donc sur la page qui liste les examens dont il est responsable (

/exams

Un responsable peut avoir accès à la liste des étudiants. Voir ceux qui n'ont rien rendu, et ceux qui ont proposé un document. Il peut alors télécharger ce qui a été proposé. (

/download/{:examid}/{:userid}

Afin de faciliter l'ordre sur l'ordinateur du responsable, les documents téléchargés ont un nom sous la forme "nomexamen\_nométudiant.ext".

```
Route : /exams/{:examid}/marks
Controller : AttendController
Action : markStudentsAction
```

### **Un utilisateur accède aux moyennes par promo**

Une page avec des statistiques simples est proposée pour tous les utilisateurs, même les non-identifiés.

```
Route : /stats
Controller : StatsController
Action : indexAction
```

### **Un utilisateur peut gérer les promos**

Il existe un utilisateur ayant le rôle d'administrateur sur le site. Il a accès à un panneau d'administration, où il va pouvoir ajouter/modifier/supprimer des promotions, voir/modifier/(dé)sactiver/changer de rôle les comptes utilisateurs.

```
Route : /admin
Controller : AdminController
Action : indexAction
```

### 0.1.3 Ce que nous n'avons pas fait

#### Génération des mots de passe

Il est demandé dans le sujet de générer aléatoirement un mot de passe lors de l'inscription d'un nouvel utilisateur, or nous utilisons un Bundle qui gère toute la partie, identification, enregistrement, récupération de mots de passe, gestion du profile... Ainsi pour redéfinir comment l'enregistrement est géré, il est nécessaire de redéfinir un certain nombre de classes (concrètement le controller, le formtype, le handler et le template). Nous avons considéré que c'était beaucoup de temps de travail pour pas grand chose donc nous nous sommes concentrés sur des tâches plus prioritaires.

### 0.1.4 3 états pour les examens

Le sujet demandait à ce qu'un examen puisse être :

- En création
- En cours
- Clos

Dans notre application, nous nous basons sur la date d'échéance de l'examen. En l'occurrence, un examen dont la date n'est pas encore passée est "en cours" tandis qu'un examen dont la date est passée est "clos" et la notation a commencé.

## 0.2 Organisation de l'équipe

Afin de travailler le plus efficacement possible, nous avons décidé d'utiliser l'outil collaboratif Github. Ceci nous a permis d'avoir chacun notre dépôt des sources du projet et de soumettre nos modifications de façon simple.

Concernant la répartition des tâches, nous avons fonctionné en déposant régulièrement des requêtes sur Github et chacun était libre de mettre en place les fonctionnalités qu'il voulait.

## 0.3 Schéma de données

Le sujet nous imposait d'utiliser plusieurs entités auxquelles nous avons ajouté un certain nombre d'attributs.

### 0.3.1 User

En plus des champs prérequis :

- Prénom
- Nom
- Adresse
- Code postal
- Ville
- Adresse e-mail

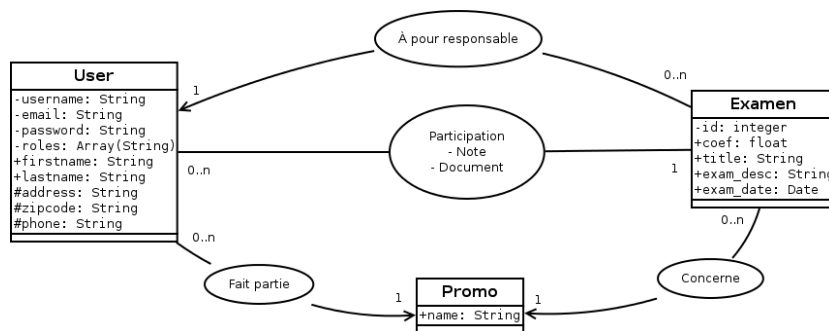


FIGURE 1 – Schéma relationnel

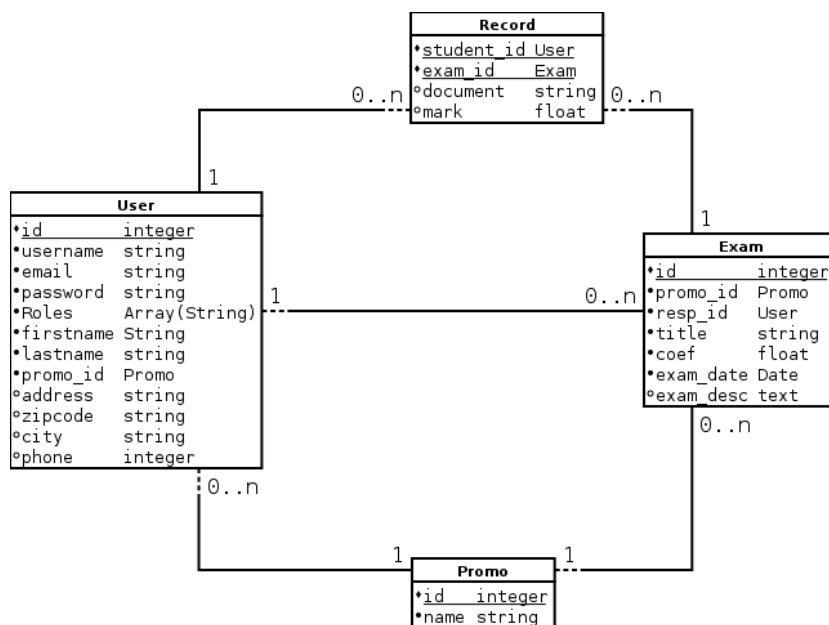


FIGURE 2 – Modèle physique de base de données

- Numéro de téléphone

Le UserBundle nous rajoute toute la partie nécessaire à la gestion de l'utilisateur côté serveur.

- Le mot de passe  $\Rightarrow$  chiffré en sha1 dans la BDD
- Les rôles  $\Rightarrow$  Pour la gestion des permissions
- Nom d'utilisateur
- Expiration du compte, confirmation par email etc.

Et enfin nous avons ajouté un champ pour notre besoin qui est une promotion, en effet, on considère qu'un étudiant appartient à une promotion précise.

### 0.3.2 Promo

Les promotions correspondent à un regroupement d'utilisateur, dans le cas d'utilisation présent, tous les étudiants d'une même année appartiennent à une promotion unique.

### 0.3.3 Exam

Un examen représente une épreuve définie par un enseignant. L'examen comprend les propriétés suivantes :

- Titre
- Description
- Promotion
- Coefficient
- Date limite
- Responsable : Un user responsable de TD

Par défaut, lors de la création de l'examen, tous les étudiants de la promotion sélectionnée sont concernés par l'examen, mais il est également possible de modifier au cas par cas si un étudiant est affecté par l'examen.

### 0.3.4 Record

L'entité Record fait le lien entre les étudiants et les examens. En effet dans un record on attribue pour un étudiant et un examen, la note, ainsi que le document rendu (Fichier pdf ou word).

On a donc :

- Étudiant
- Examen
- Note
- Document

Ainsi un Record avec la note et le document NULL, correspond au fait qu'un étudiant doit effectuer un rendu pour un tel examen. Quand il rend quelque chose, on peuple le champ 'Document'. Et ensuite quand l'examen est terminé le responsable de l'examen peut associer une note.



## **0.4 Réalisation Technique**

### **0.4.1 UserBundle**

Nous avons utilisé un Bundle d’extension nommé : UserBundle.

Ce bundle constitue la brique applicative qui permet d’effectuer toutes les actions nécessaires à presque tous les projets, c’est-à-dire :

- Inscription
- Authentification
- Connexion
- Déconnexion
- Gestion du profil
- Changement de mot de passe

Nous évitant du travail laborieux qui ne sert pas sur le plan métier de l’application.

### **0.4.2 Les tests avec PHPUnit**

Nous avons principalement testé les différentes entités. Grâce à PHPUnit nous avons généré la couverture en tests de notre code.

Couverture des tests

### **0.4.3 La documentation développeur avec phpDocumentor**

Tout notre code a été documenté en utilisant phpDocumentor. On peut trouvé cette documentation à l’adresse :

Documentation développeur

## **Conclusion**