

---

# **Conception d'une plateforme de e-learning**

## **Développement du tableau de bord professeur**

*Travail de maturité, Collège du Sud*

**Bryan Oberson**

30 mars 2015



<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Fonctionnalités du tableau de bord</b>	<b>5</b>
2.1	Ajouter un groupe . . . . .	5
2.2	Gérer un groupe . . . . .	5
2.3	Voir ses exercices . . . . .	7
2.4	Changer de mot de passe . . . . .	8
<b>3</b>	<b>Guide du développeur</b>	<b>11</b>
3.1	Démarrer Django dans un workspace Cloud9 . . . . .	11
3.2	Navigation . . . . .	12
3.3	Use Cases . . . . .	12
3.4	Dossier <code>static</code> . . . . .	12
3.5	Gabarits . . . . .	13
3.6	Fichiers importants . . . . .	14
3.7	Modèles . . . . .	14
3.8	Vues . . . . .	15
3.9	Urls . . . . .	16
<b>4</b>	<b>Développement dirigé par les tests</b>	<b>17</b>
4.1	Les tests fonctionnels . . . . .	17
4.2	Les tests unitaires . . . . .	17
4.3	Les tests d’approbation . . . . .	17
4.4	Le cycle du développement dirigé par les tests . . . . .	18
4.5	Gain de temps ? . . . . .	18
<b>5</b>	<b>Conclusion</b>	<b>19</b>
<b>6</b>	<b>Sources</b>	<b>21</b>
6.1	Bibliographie . . . . .	21
6.2	Webographie . . . . .	21
<b>7</b>	<b>Annexes</b>	<b>23</b>
7.1	Déclaration personnelle . . . . .	23
7.2	Remerciements . . . . .	23
7.3	Code source . . . . .	24
<b>8</b>	<b>Table des illustrations</b>	<b>31</b>



Ce travail de maturité a été réalisé sous la direction de M. Donner

---

# Introduction

---

Ce document est la documentation du tableau de bord destiné à être utilisé par les professeurs pour un site web d'e-learning pour les mathématiques.

La problématique de ce travail s'intitule : «Développement du tableau de bord professeur». Il a été nécessaire de faire attention à ne pas dépasser sur les applications des différents contributeurs à ce projet étant donné que le tableau de bord sert de centre aux différentes applications du site.

Le dessin de cette documentation sera tout d'abord d'expliquer les fonctionnalités de ce tableau de bord et de les illustrer. Cela permettra à un professeur néophyte au tableau de bord de bien comprendre comment le prendre en main.

De plus, la documentation expliquera le côté programmation de l'application. En effet, cela comportera entre autre l'explication des différents fichiers, des schémas résumant certaines caractéristiques de l'application ainsi que les étapes à suivre pour démarrer un serveur Django depuis Cloud9.

La dernière partie consistera en une approche au développement dirigé par les tests. Elle sera composée d'une explication des termes importants ainsi que de l'explication du cycle fondamental dudit développement

Dans le projet final, qui consiste donc en un site web pour apprendre les mathématiques, le tableau de bord sera le centre de toute activité. En effet, c'est depuis cette application que les groupes seront créés, gérés et que les exercices, quiz et cours seront gérés. Sans cette application, il ne serait pas possible d'avoir une supervision des élèves ainsi qu'un cadre pour les aider à s'améliorer.

Par exemple, pour aider différents élèves possédant les mêmes problèmes dans les mêmes sujets, le professeur pourra créer un groupe dédié à ces étudiants et pourra leur assigner des devoirs plus spécifiques pour les aider à progresser. Cette application s'avérera donc être une grande aide pour les professeurs afin qu'ils puissent mieux gérer les différents niveaux de leurs élèves.

Dans l'enseignement, il est plus difficile d'aider chaque élève selon leurs difficultés et leurs aises. Cette application permettra donc un enseignement plus précis pour que chaque élève puisse trouver ce qu'il lui plaise.



---

## Fonctionnalités du tableau de bord

---

La première partie de cette documentation consiste en une explication des différentes fonctionnalités du tableau de bord.

Il est important qu'un professeur puisse prendre correctement l'application en main dès le début car cela lui permettra de l'utiliser à meilleur escient et de ne pas passer à côté d'une fonctionnalité ou encore de perdre du temps à comprendre son utilisation.

### 2.1 Ajouter un groupe

La fonctionnalité de base de ce tableau de bord est la création de groupe. En créant un groupe, le professeur sera par la suite capable de le gérer en supervisant les membres qui s'y trouvent mais aussi en y assignant des devoirs.

Pour créer un groupe, il suffit de se rendre sur Nouveau groupe, tout en bas dans le menu de gauche. Cette action fera apparaître le formulaire de création de groupe.

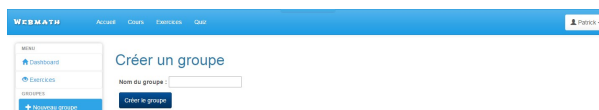
The screenshot shows the 'Créer un groupe' (Create a group) form in the WEBMATH application. On the left is a sidebar menu with options: 'Accueil', 'Créer', 'Gérer', 'Paramètres', and 'Ajouter un groupe' (highlighted). The main content area is titled 'Créer un groupe' and contains a text input field labeled 'Nom du groupe' and a blue 'Créer le groupe' button.

FIGURE 2.1 – Formulaire de création de groupe

La seule exigence présente lors de la création d'un groupe est le nom. Une fois le groupe créé, l'utilisateur actuel est automatiquement défini en tant que professeur pour le groupe.

Le groupe précédemment créé sera désormais affiché en permanence dans le menu de gauche du professeur, ce qui lui permet d'accéder à ses informations quand il le désire.

### 2.2 Gérer un groupe

#### 2.2.1 Gérer les membres du groupe

Depuis cette page, le professeur peut gérer les membres qui sont actuellement enregistrés dans le groupe.



Il peut tout d'abord rajouter les élèves ou professeurs qu'il souhaite en entrant leur nom d'utilisateur dans le champ à disposition.

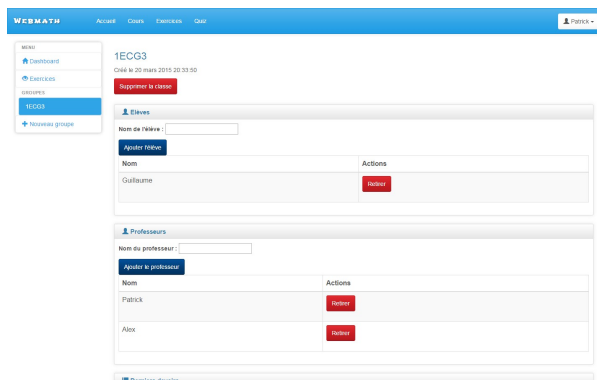


FIGURE 2.2 – Page d'administration d'un groupe

Si le nom d'utilisateur rentré correspond bien à un étudiant ou à un professeur, cet utilisateur sera rajouté dans la liste des membres.

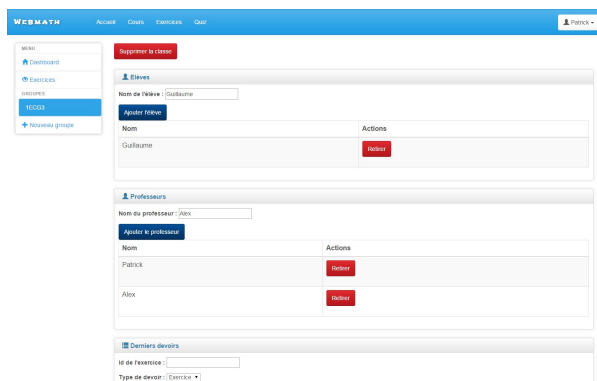


FIGURE 2.3 – Ce à quoi ressemble la page une fois que des membres ont été rajoutés

Au contraire, si aucun utilisateur n'a été trouvé ou si l'utilisateur ne correspond pas au rôle qui lui est donné (par exemple si c'est un professeur et qu'il a été ajouté aux étudiants), un message d'erreur sera retourné.

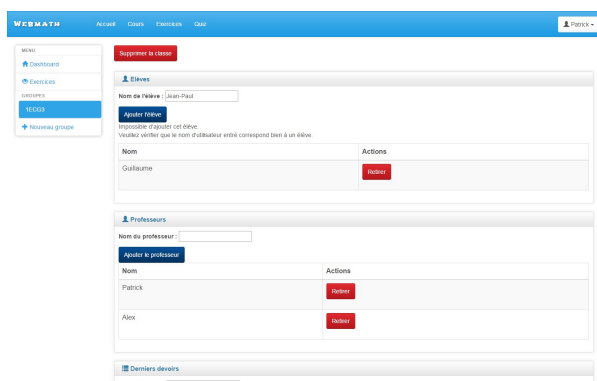


FIGURE 2.4 – Message d'erreur retourné si l'utilisateur n'est pas valable

Une fois ajouté, un membre peut facilement être retiré du groupe grâce au bouton Retirer qui se trouve à côté de son nom.

## 2.2.2 Gérer un devoir

Un professeur peut bien évidemment donner des devoirs à un groupe.

Un devoir peut être un exercice, un quiz ou un cours, et avoir été créé par le professeur actuellement en ligne ou un autre.

Pour assigner un devoir, il suffit de savoir le numéro de l'exercice, quiz ou cours, et de préciser grâce au menu à choix de quel type d'activité il s'agit.

Nombre de l'exercice	Type de devoir	Actions
Equation du deuxième degré	Exercice	Retirer
Factorisation	Exercice	Retirer
Courte équation	Quiz	Retirer
Nombre de Villes	Cours	Retirer

FIGURE 2.5 – Différents champs à compléter pour assigner un devoir

Comme pour les fonctionnalités précédentes, si aucun exercice, quiz ou cours n'a pu être associé au numéro entré, un message d'erreur sera renvoyé.

Impossible d'assigner ce devoir.  
Veuillez vérifier que l'exercice, quiz ou cours correspondant à cet id existe réellement.

Nombre de l'exercice	Type de devoir	Actions
Equation du deuxième degré	Exercice	Retirer
Factorisation	Exercice	Retirer
Courte équation	Quiz	Retirer
Nombre de Villes	Cours	Retirer

FIGURE 2.6 – Message d'erreur retourné si l'activité n'a pas pu être trouvée

Un devoir peut être à tout moment retiré grâce au bouton Retirer à sa droite.

## 2.3 Voir ses exercices

Dans le menu de gauche, il y a un bouton nommé Exercices. C'est depuis cette page que le professeur pourra voir ses exercices, ses quiz et ses cours.

Pour chaque activité que le professeur aura créée, il pourra voir le titre qu'il lui a donné, la date à laquelle il l'a créée et son numéro qui lui sera utile s'il veut l'assigner en tant que devoir à l'un de ses groupes.

Il peut bien évidemment supprimer une activité en utilisant le bouton Supprimer se trouvant dans la dernière colonne du tableau.

Si le professeur souhaite créer une nouvelle activité, il n'a qu'à utiliser le bouton Créer en haut du tableau qui le redirigera directement au formulaire de création.

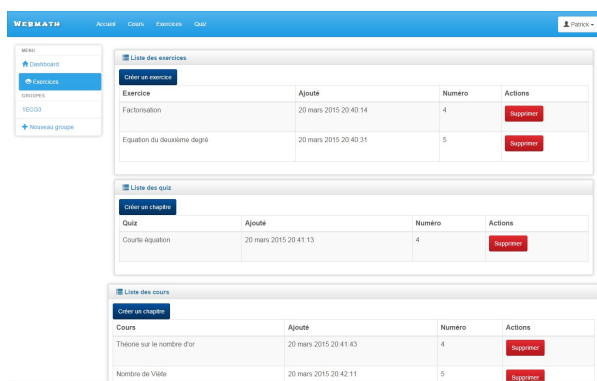


FIGURE 2.7 – Ce à quoi ressemble la page Exercices

## 2.4 Changer de mot de passe

Peu importe sur quelle page il se trouve, le professeur peut accéder à un menu déroulant en haut à droite de cette page.

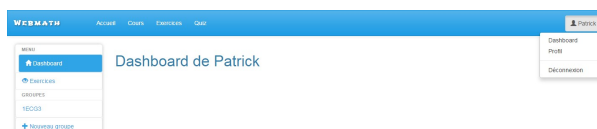


FIGURE 2.8 – Apparence du menu déroulant

Dashboard amène le professeur sur l'accueil de son tableau de bord, Déconnexion le déconnecte et Profil l'amène sur un formulaire de changement de mot de passe.

Pour le modifier, le professeur n'a qu'à remplir les deux champs et à valider. Si tout a été rentré correctement, le mot de passe sera correctement modifié.



FIGURE 2.9 – Message pour confirmer que le changement de mot de passe a correctement eu lieu

Au contraire, s'il y a une erreur, un message pour prévenir le professeur sera retourné.



FIGURE 2.10 – Message d’erreur retourné si les champs n’ont pas correctement été remplis



---

## Guide du développeur

---

La deuxième partie de cette documentation consiste en une explication du fonctionnement de l'application du côté du développeur.

Il est donc expliqué comment les vues, modèles ou urls de cette application fonctionnent et l'utilité de certains fichiers. Il y a aussi des schémas permettant de mieux comprendre le fonctionnement du tableau de bord.

### 3.1 Démarrer Django dans un workspace Cloud9

Cloud9 est un éditeur de code en ligne permettant de programmer depuis n'importe où. Cette plateforme a été utilisé pour le développement de l'application.

Voici les quelques étapes à suivre pour démarrer Django dans Cloud9<sup>1 2</sup>.

Il faut tout d'abord créer un workspace.

**Attention, si on souhaite utiliser Python3 il est conseillé de créer un workspace de type Custom, comme sur la photo suivante :**

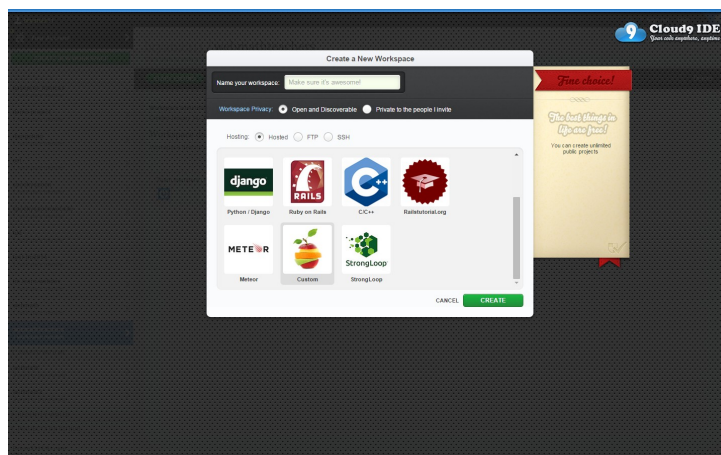


FIGURE 3.1 – Créer un workspace de type Custom

Une fois sur le dépôt, il faut tout d'abord installer Django avec la commande suivante :

```
sudo pip3 install django==1.7
```

Django installé, il faut démarrer un projet avec la commande suivante :

1. «Cloud9 - Your development environment, in the cloud», consulté le 29.03.2015, <https://c9.io/>
2. «Configuration de Django 1.7 sous Cloud9», consulté le 24.03.2015, <http://www.donner-online.ch/webtutos/django/c9config.html>

```
django-admin startproject nom-pour-le-projet
```

Maintenant que le projet existe, il est possible de créer autant d'applications que souhaité avec la commande :

```
python3 manage.py startapp nom-pour-l'application
```

Pour lancer le serveur, il est nécessaire de taper cette commande :

```
python3 manage.py runserver $IP:$PORT
```

### 3.2 Navigation

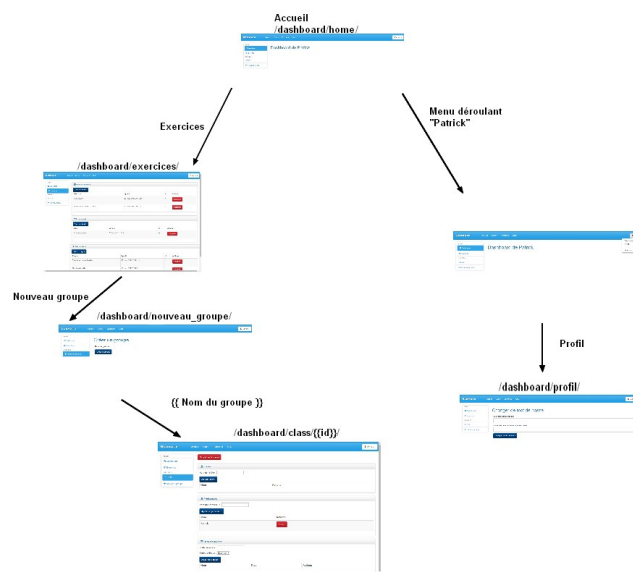


FIGURE 3.2 – Schéma de navigation du site

Ce schéma explique les relations qui existent entre les différentes pages. Plus précisément, comment accéder une page depuis une autre.

Il est important de noter que le menu déroulant ainsi que les pages Exercices, Nouveau groupe et la page d'une classe peuvent être atteintes depuis n'importe quelle page du tableau de bord.

### 3.3 Use Cases

Ce schéma explique les différentes actions qui se passent lorsque le professeur veut utiliser une des fonctionnalités du tableau de bord.

Par exemple, s'il veut ajouter un élève dans un de ses groupes, il n'a qu'à entrer son nom, le serveur ira le chercher et l'ajoutera dans le groupe.

### 3.4 Dossier static

Le dossier static est utilisé pour garder tous les fichiers tel que les fichiers CSS ou les fichiers Javascript. Dans cette application, il contient les dossiers suivants :

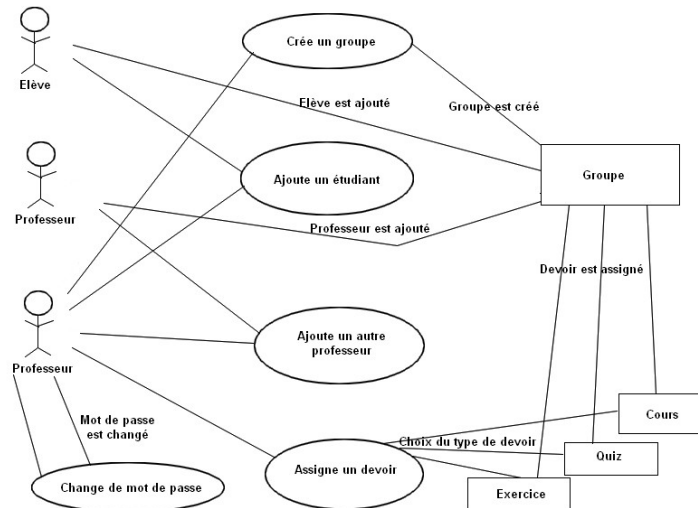


FIGURE 3.3 – Schéma résumant les actions qui se déroulent selon les utilisations du professeur

- `bower_components` : ce dossier contient tous les éléments du front-end qui possèdent des dépendances, comme les fichiers `bootstrap` ou des fichiers de base pour `jquery`. Le dossier `bower_components` contient les fichiers relatifs au thème Bootstrap utilisé. Plus précisément, il contient le `css` et le `javascript`.
- `css` : dans ce dossier se trouvent tous les fichiers `css` qui sont nécessaires pour le design du site. La différence entre les fichiers qui se trouvent dans ce dossier et les fichiers `css` du dossier `bower_components` est que les premiers servent de base et ne sont pas adaptatifs alors que les derniers permettent les changements de place et de taille que nous offre l'adaptivité de Bootstrap.
- `fonts` : le dossier `fonts` de l'application tableau de bord contient toutes les informations relatives aux petits signes (glyphicons) qui sont utilisés dans le tableau de bord, comme le + devant «Nouveau groupe».

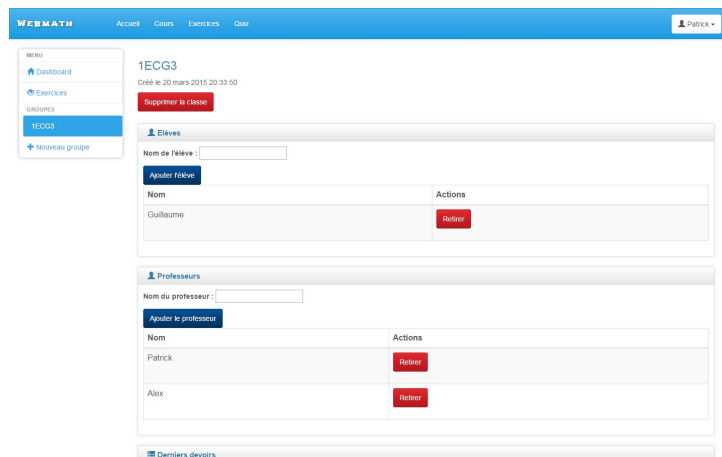


FIGURE 3.4 – Exemples de glyphicons dans le menu de gauche : l'oeil ou la maison

## 3.5 Gabarits

Dans le dossier `templates` se trouvent tous les fichiers `html` servant de gabarits à l'application :

- Le gabarit `classe.html` contient le gabarit utilisé pour l'affichage des groupes. Il affiche tout d'abord le nom du groupe ainsi que la date de sa création, puis chaque élève et professeur ainsi qu'un bouton pour les retirer du groupe. Dans les tableaux affichant les élèves et professeur est aussi affiché un bouton pour



rajouter des membres. S'il est impossible d'ajouter le membre dont le nom a été rentré, le gabarit retourne un message d'erreur.

Il affiche enfin les devoirs par type d'activité (exercice, quiz, cours) et un bouton pour les retirer. Il y a aussi un bouton pour assigner des devoirs. De nouveau, si aucune activité n'a été trouvée, le gabarit retourne un message d'erreur.

- Le gabarit `newclass.html`, lui, sert à créer un nouveau groupe qui pourra ensuite être supervisé. Il ne fait qu'afficher un champ pour le nom et un bouton de confirmation. Une fois le groupe créé, un message de confirmation est retourné.
- Le gabarit `index.html` ne contient pour le moment que le nom d'utilisateur du professeur actuellement connecté. Il contiendra plus tard des statistiques quant aux groupes ou aux activités du professeur.
- Le gabarit `exercices.html` affiche les exercices, les quiz et les cours qui ont été créés par le professeur. Ces activités sont supprimables depuis cette page et le professeur a la possibilité d'accéder aux formulaires de création d'activité. Il peut aussi voir quand ces activités ont été créées.
- Le gabarit `profile.html` sert au changement de mot de passe. Il affiche deux champs qui doivent être remplis de façon identique. Si le changement de mot de passe a bien pu avoir lieu, un message de confirmation est retourné. Dans le cas contraire, un message d'erreur s'affiche.

Ces 5 gabarits ont tous la même structure de base :

- Une bande au sommet de la page qui, une fois les applications ensemble, amènera l'utilisateur à ces applications.  
Il y a aussi un menu déroulant permettant d'accéder au tableau de bord, au changement de mot de passe et permettant aussi à l'utilisateur de se déconnecter.
- Un menu à gauche de la page permettant d'accéder aux exercices, aux différents groupes qui sont tous ajoutés en liste et à l'option de création de groupe.

## 3.6 Fichiers importants

Les applications Django possèdent les fichiers de base suivants :

- `models.py` qui est utilisé pour créer les différents modèles et leur attribuer des champs.
- `admin.py` est utilisé pour signaler à Django quels sont les modèles qui doivent apparaître dans l'application admin. Une fois qu'ils y apparaissent, il est possible de créer, modifier ou supprimer n'importe quel objet depuis cette application.
- Le fichier `forms.py` est celui dans lequel on peut entrer les différents formulaires dont l'on a besoin pour l'application.
- C'est dans `views.py` que l'on peut stocker des variables nécessaires dans certains gabarits, mais aussi réaliser certaines actions comme la suppression d'un objet. A la fin d'une vue, on retourne souvent un fichier `html` ou on redirige vers une autre vue.
- Le fichier `urls.py` contient les informations concernant les différentes urls accessibles par l'utilisateur et quelles vues sont censées être utilisées.

### 3.6.1 Fichiers uniques de Django

On peut modifier le fichier `settings.py` afin de définir la zone temporelle dans laquelle on se trouve, mais aussi les applications qu'un projet doit gérer ou encore l'emplacement du fichier `static`. Il sert donc de configuration de base pour un projet.

Il y a aussi un autre fichier `urls.py` qui, lui, est très utile si l'on doit s'occuper de plusieurs applications à la fois. En effet, on peut définir le début de l'url et rediriger vers un autre fichier `urls.py`.

## 3.7 Modèles

### 3.7.1 Modèles utilisés pour le tableau de bord

Il y a tout d'abord le modèle `BaseProfile` qui découle de `User` et qui, comme son nom l'indique, va servir de profil de base pour le modèle `Teacher` et `Student`.

L'utilisateur Django possède de base les caractéristiques suivantes <sup>3</sup> :

- username : nom d'utilisateur
- first\_name : prénom
- last\_name : nom
- email : adresse courriel
- password : mot de passe
- group : les relations avec le modèle Group de Django
- user\_permissions : les relations avec le modèle Permission de Django
- is\_staff : si l'utilisateur peut accéder à l'application admin
- is\_active : définit si l'utilisateur doit être considéré comme actif ou non
- is\_superuser : définit si l'utilisateur a tous les droits
- last\_login : dernière connexion de l'utilisateur
- date\_joined : date de création de l'utilisateur

Car un professeur a besoin de voir ses exercices, quiz et cours, et pourra les assigner en tant que devoirs à un groupe, les modèles Exercise, Quiz et Course ont tous les trois été apportés.

Il y a ensuite le modèle Group, qui n'est pas le même que celui implémenté de base avec Django, car c'est celui qui a été utilisé pour les groupes d'un professeur. Les membres sont ajoutés par le biais du modèle Groupmembers qui sert de table intermédiaire entre Student ainsi que Teacher et Group. Le modèle AssignHomework, qui est aussi une table intermédiaire, sert à l'affectation de devoirs entre Exercise, Quiz, Course et Group.

### 3.7.2 Diagramme UML

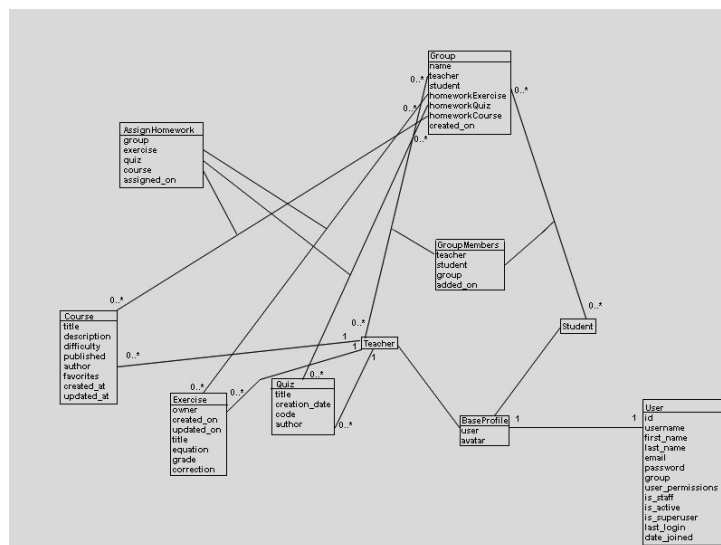


FIGURE 3.5 – Schéma résumant les relations entre les différents modèles

Sur ce schéma, les types de lien existant entre les différents modèles sont plus visibles :

- 0..\* signifie qu'un objet peut posséder entre zéro et l'infini objets appartenant à l'autre modèle
- 1 signifie qu'il ne peut en posséder qu'un seul

Il est aussi facile de voir la place que prennent les tables intermédiaires AssignHomework et GroupMembers : elles servent de ponts entre deux modèles.

## 3.8 Vues

Toutes les vues vont devoir chercher le professeur correspondant à l'utilisateur actuellement connecté. Cela permettra à chaque fois d'aller chercher les données correspondantes comme le nom d'utilisateur toujours affiché sur le menu déroulant en haut à droite de la page.

3. «django.contrib.auth», consulté le 23.03.2015, <https://docs.djangoproject.com/en/1.7/ref/contrib/auth/>

La vue `home` sert uniquement à distinguer la première lettre du nom d'utilisateur pour qu'apparaisse dans le gabarit «de» ou «d'».

La vue `exercices`, elle, ne cherche rien de plus. L'utilisateur nous permettra d'accéder aux exercices, quiz et cours qui lui sont associés mais tout ceci est directement recherché dans le gabarit.

C'est grâce à la vue `newgroup` qu'un professeur peut créer un groupe. S'il veut créer un groupe, la vue se contentera de créer un groupe associé au nom et de créer un lien entre le professeur et le groupe grâce à la table intermédiaire `GroupMembers`. La variable `success` a pour utilité d'afficher un message de confirmation dans le gabarit `newclass.html` une fois le groupe correctement créé.

La vue `profil` est celle utilisée pour le changement de mot de passe. Elle compare les deux mots de passe entrés. Si les deux mots de passe correspondent, le mot de passe est attribué à l'utilisateur et, grâce au gabarit `profile.html` et à la variable `success`, un message est retourné pour confirmer le changement. Dans le cas contraire, un message d'erreur est retourné.

La vue `groupe`, elle, est composée de plusieurs actions qui dépendent de la forme qui a été remplie.

- Il y a tout d'abord `addTeacher` qui, quand l'utilisateur entre le nom d'utilisateur d'un autre professeur pour l'ajouter dans un groupe existant, va créer un objet `GroupMembers` entre ce professeur et le groupe actuel pour qu'il fasse parti de ce groupe. Il se passe la même chose pour `addStudent` si le même utilisateur décide d'ajouter un élève.
- Pour assigner un devoir à un groupe, la vue va utiliser `assignHomework` qui, selon le genre d'activité et le numéro qui ont été sélectionnés par l'utilisateur, va chercher l'activité et créer un objet `AssignHomework` qui va lier l'exercice, le quiz ou le cours au groupe.
- Pour supprimer un groupe, il y a d'abord l'utilisation de `deleteClass` qui va uniquement servir à l'apparition d'un deuxième bouton qui activera `deleteClassConfirm`, qui supprimera le groupe et donc tous les objets `AssignHomework` et `GroupMembers` avec lesquels il était associé.

Cette vue va par la suite retourner le gabarit `classe.html` avec les variables définies au début qui apparaîtront sur la page.

Finalement, quelques vues ont été réalisées pour des actions plus complexes. Par exemple, `deleteFromGroup` avait besoin de deux variables, `member_id` et `group_id`. Cette vue a donc été liée à une url nécessitant ces deux variables. La vue `deleteFromGroup`, composée de `deleteStudent` et `deleteTeacher`, servent à retirer les membres d'un groupe en supprimant l'objet `GroupMembers` qui les liait. La vue `deleteActivity`, qui elle est composée de `deleteExercise`, `deleteQuiz` et `deleteCourse` sert à supprimer une activité depuis son tableau de bord. Enfin, `deleteHomework` permet au professeur de retirer un devoir précédemment assigné selon le type d'activité auquel il correspond.

### 3.9 Urls

Les urls `home`, `group_view`, `exercices`, `newgroup` et `profil` redirigent simplement aux vues du même nom.

Les urls `deleteFromGroup`, `deleteActivity` et `deleteHomework`, elles, sont reliées aux vues du même nom qui permettent certaines actions dépendantes de variables très précises. Pour réaliser ceci, des formes ont été créées dans les gabarits redirigeant à ces urls et possédant les variables nécessaires afin que le programme puisse aller chercher les objets souhaités et permettre, par exemple, la suppression d'une activité.

#### Note de bas de page

---

## Développement dirigé par les tests

---

Le développement dirigé par les tests, grossièrement traduit de l'anglais Test Driven Development, est une technique de développement utilisée par beaucoup de programmeurs.

En effet, c'est grâce à cette technique que l'on peut le mieux s'assurer de la fonctionnalité du site et de la simplicité du code. Dans un travail de longue haleine, cette méthode devient nécessaire pour ne pas être redondant dans son code et pour le rendre le plus clair possible.

Dans cette partie, nous allons nous intéresser au côté théorique de ce type de développement.

Le développement dirigé par les tests est composé de deux tests importants et bien différents, même s'il existe un troisième qui est utilisé dans un contexte très précis.

### 4.1 Les tests fonctionnels

Le test fonctionnel, comme son nom l'indique, cherche à tester la fonctionnalité du site. Plus précisément, il se met du côté de l'utilisateur du site. Il va par exemple vérifier que les titres et les textes apparaissent, mais il va aussi regarder si les différents boutons ou champs de textes fonctionnent <sup>1</sup>.

En général, on remplit ces tests de commentaires qui racontent une histoire pour pouvoir s'y retrouver plus facilement quand on le lit. Par exemple, on pourrait raconter l'histoire d'un professeur qui découvre un site web d'e-learning pour les mathématiques et qui d'abord crée un compte, puis essaie de créer une classe.

### 4.2 Les tests unitaires

Le test unitaire, lui, se base plus sur le point de vue du programmeur. Si on pouvait considérer le test fonctionnel comme un test externe, le test unitaire, lui, serait le test interne. Ce qu'il teste ne sera jamais vu, car il permet de vérifier que le code fonctionne comme prévu <sup>1</sup>.

### 4.3 Les tests d'approbation

Plus rare, il existe aussi le test d'approbation qui peut même amener à un développement plus précis : Acceptance Test Driven Development (ATDD), ou développement dirigé par les tests d'approbation <sup>2</sup>.

Le grand avantage de ces tests est qu'ils permettent un meilleur travail de groupe. En effet, l'équipe travaillant sur le projet se réunit et décide des critères de base de ce projet qui seront ensuite écrits en test. Par conséquent, toute l'équipe est d'accord sur ces critères et il sera ensuite facile de voir s'ils sont respectés.

---

1. PERCIVAL, Harry J.W., «Test Driven Development With Python», publié le 19 juin 2014

2. «Acceptance Test Driven Development (ATDD) : An Overview», consulté le 25.03.2015, <http://testobsessed.com/2008/12/acceptance-test-driven-development-atdd-an-overview/>

## 4.4 Le cycle du développement dirigé par les tests

Quand on veut programmer à l'aide du développement dirigé par les tests, on tente de suivre un certain cycle <sup>1</sup> :

1. Tout d'abord, il est **impératif** d'écrire un test avant même d'écrire n'importe quelle ligne de code. En effet, l'idée du développement dirigé par les tests est d'être sûr qu'un test échoue avant d'écrire notre code. Le test peut être fonctionnel, unitaire, ou d'approbation (dans le cadre du ATDD), cela dépendant de la partie de l'application que l'on souhaite développer. Le test va évidemment retourner un message négatif, mais c'est normal étant donné que rien n'a été codé concernant la fonctionnalité testée.
2. Ensuite seulement, le but est d'écrire un minimum de code possible pour que le test précédemment lancé fonctionne. Il faut donc réussir à ce que le test, une fois relancé, retourne un résultat positif. Il faut faire attention à ne pas développer une fonctionnalité qui n'est pas dans le test
3. Une fois le code écrit, on peut relancer ce test. Si le résultat est positif, on peut passer à l'étape suivante. Dans le cas contraire, il faudra refaire les étapes 2 et 3 jusqu'à ce que le test fonctionne.
4. Finalement, il ne reste qu'à restructurer le code précédemment écrit pour qu'il soit plus lisible. Il faut faire très attention durant cette étape à ne rien ajouter ou enlever. Le code doit garder le même résultat.

Une fois que ces 4 étapes ont été effectuées, il ne reste qu'à recommencer avec une autre fonctionnalité de l'application, jusqu'à que celle-ci soit finie.

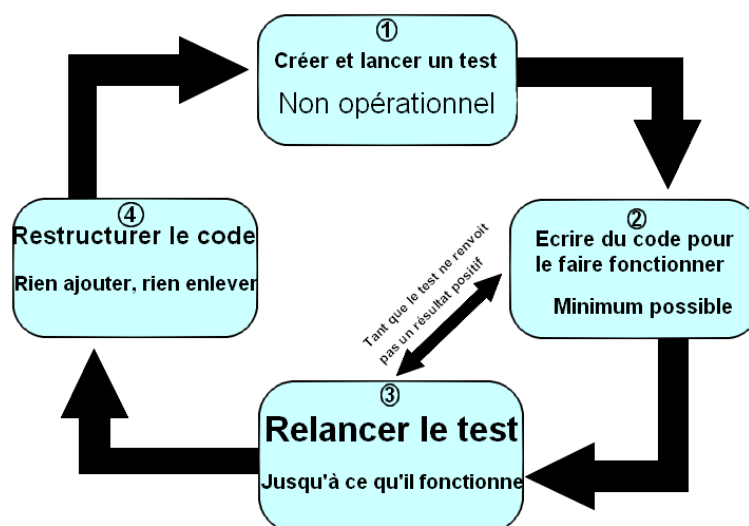


FIGURE 4.1 – Schéma résumant les 4 étapes du développement dirigé par les tests

## 4.5 Gain de temps ?

En lisant ces 4 étapes répétitives, on ne peut que se demander si le Test Driven Development et son cycle compliqué est réellement un atout et un gain de temps pour le programmeur.

Il est clair que, sur un travail de petite taille, tout tester n'aurait pas énormément de sens, car tout peut être facilement essayable par soi-même. Dans le cas d'un travail d'une certaine consistance, ce n'est pas pareil. C'est uniquement en testant que l'on peut être sûr de son code, car cela signifie qu'il est valide et devrait le rester.

Note de bas de page

---

### Conclusion

---

En conclusion, si l'on pose un regard critique sur le tableau de bord destiné aux professeurs présenté dans cette documentation, nous pouvons voir qu'il possède des limites.

En effet, bien que la création de groupe soit un outil très utile, la faculté de voir les statistiques de ses élèves ou encore les résultats aux différentes activités serait un bien incommensurable pour que le professeur puisse s'adapter aux besoins des étudiants et puisse leur proposer des exercices adaptés. De plus, il aurait été utile que le professeur puisse observer les résultats aux exercices de certains élèves.

A l'avenir, il serait envisageable de pouvoir accéder à un tableau mettant en valeur les résultats obtenus par les élèves à des activités précises, permettant de comparer et d'aider les élèves en difficulté avant un examen ou encore de les ajouter à un groupe d'étudiants aux mêmes difficultés.



---

**Sources**

---

## **6.1 Bibliographie**

ALEXIS, Pierre et BERSINI, Hugues, «Apprendre la programmation web avec Python et Django», chapitres 5 et 10, publié le 9 novembre 2012

PERCIVAL, Harry J.W., «Test Driven Development With Python», chapitres 1 à 9, publié le 19 juin 2014

## **6.2 Webographie**

«Dash | Learn HTML, CSS, JavaScript with our free online tutorial | General Assembly», consulté le 16.08.2014, <https://dash.generalassemb.ly/>

«France-IOI - Cours et problèmes», consulté le 29.08.2014, <http://www.france-ioi.org/algo/chapters.php>

«django.contrib.auth», consulté le 23.03.2015, <https://docs.djangoproject.com/en/1.7/ref/contrib/auth/>

«Configuration de Django 1.7 sous Cloud9», consulté le 24.03.2015, <http://www.donner-online.ch/webtutos/django/c9config.html>

«Acceptance Test Driven Development (ATDD) : An Overview», consulté le 25.03.2015, <http://testobsessed.com/2008/12/acceptance-test-driven-development-atdd-an-overview/>

«Cloud9 - Your development environment, in the cloud», consulté le 29.03.2015, <https://c9.io/>





---

## Annexes

---

### 7.1 Déclaration personnelle

Nom : Oberson

Prénom : Bryan

Adresse : Route du Lac Lussy 3, 1618 Châtel-St-Denis

1. Je certifie que le travail **Développement du tableau de bord professeur** a été réalisé par moi conformément au Guide de travail des collèves et aux Lignes directrices de la DICS concernant la réalisation du Travail de maturité.

2. Je prends connaissance que mon travail sera soumis à une vérification de la mention correcte et complète de ses sources, au moyen d'un logiciel de détection de plagiat. Pour assurer ma protection, ce logiciel sera également utilisé pour comparer mon travail avec des travaux écrits remis ultérieurement, afin d'éviter des copies et de protéger mon droit d'auteur. En cas de soupçon d'atteintes à mon droit d'auteur, je donne mon accord à la direction de l'école pour l'utilisation de mon travail comme moyen de preuve.

3. Je m'engage à ne pas rendre public mon travail avant l'évaluation finale.

4. Je m'engage à respecter la Procédure d'archivage des travaux de maturité/travaux personnels/travaux de maturité spécialisée en vigueur dans mon école.

5. J'autorise la consultation de mon travail par des tierces personnes à des fins pédagogiques et/ou d'information interne à l'école :

— oui

— non (car il contient des données personnelles et sensibles.)

Lieu, date : \_\_\_\_\_

Signature : \_\_\_\_\_

### 7.2 Remerciements

Je tiens à remercier M. Cédric Donner, superviseur du séminaire, pour sa disponibilité à tout moment, ses conseils quant au travail ainsi que son encouragement tout au long de ce travail.

Je remercie Keran Kocher (3GY3), Benoît Léo Maillard (3GY1), Florian Genilloud (3GY2), Daniel Nunes Silva (3GY1) et Sébastien Chuat (3GY3), qui ont aussi travaillé sur ce projet, pour leurs réponses à mes questions et leur entraide dans les sujets difficiles.

Je remercie aussi Léa Gobet (3GY3) pour m'avoir aidé quant à la mise en forme de mon travail.

Merci encore à Alexandre Currat (3GY7) pour avoir essayé l'application et m'avoir prévenu de tous les bogues présents ainsi que les améliorations envisageables.

## 7.3 Code source

### 7.3.1 models.py

```
1  from django.db import models
2  from django.contrib.auth.models import User
3
4
5  #Profile de base découlant de User
6
7  class BaseProfile(models.Model):
8      user = models.OneToOneField(User) #Donne les attributs de User à BaseProfile
9      avatar = models.ImageField(null=True, blank=True, upload_to="avatars/")
10
11     class Meta:
12         abstract = True
13
14
15     #Les deux modèles héritent de BaseProfile et donc de User
16
17     class Teacher(BaseProfile):
18
19         def __str__(self):
20             return "Professeur {}".format(self.user.username)
21
22     class Student(BaseProfile):
23
24         def __str__(self):
25             return "Etudiant {}".format(self.user.username)
26
27
28     #
29     # Modèle de Keran pour les cours
30     #
31
32     class Course(models.Model):
33         title = models.CharField(max_length=30, unique=True)
34         description = models.TextField()
35         difficulty = models.IntegerField()
36         published = models.BooleanField(default=False)
37
38         author = models.ForeignKey(Teacher)
39         #chapter = models.ForeignKey('teachers.Chapter', related_name="courses")
40         favorites = models.ManyToManyField(User, related_name="favorite_courses", blank=True, null=True)
41         # videos = models.ManyToManyField(Video)
42         # images = models.ManyToManyField(Image)
43         # definitions = models.ManyToManyField(Definition)
44
45         created_at = models.DateTimeField(auto_now_add=True)
46         updated_at = models.DateTimeField(auto_now=True)
47
48         def __str__(self):
49             return self.title
50
51     #
52     # Modèle de Florian pour les exercices
53     #
54
55     class Exercise(models.Model):
56
57         owner = models.ForeignKey(Teacher) # créateur de l'exercice
58         created_on = models.DateTimeField(auto_now_add=True) # Date de création
```

```

59     updated_on = models.DateTimeField(auto_now=True)
60     title = models.CharField(max_length=30) # C'est le titre de l'exercice
61     equation = models.CharField(max_length=50) # C'est l'équation entrée par le professeur
62     grade = models.CharField(max_length=60) # donnée une note de difficulté à l'exercice
63     correction = models.CharField(max_length = 200) # corrigé de l'exercice ( obligatoire )
64     def __str__(self):
65         return self.title
66
67 #
68 # Modèle de Benoit pour les quiz
69 #
70
71 class Quiz(models.Model): #Infos générales sur le quiz
72     title = models.CharField(max_length=100)
73     creation_date = models.DateTimeField(auto_now_add=True)
74     code = models.CharField(max_length=1000) #Format texte du quiz
75     author = models.ForeignKey(Teacher)
76     #id_chapter = models.ForeignKey('teachers.Chapter')
77
78     def __str__(self):
79         return self.title
80
81
82
83
84 #Modèle pour les groupes
85 class Group(models.Model):
86     name = models.CharField(max_length=30)
87     teacher = models.ManyToManyField(Teacher, through='GroupMembers')
88     student = models.ManyToManyField(Student, through = 'GroupMembers')
89     #uniquement les devoirs exercices
90     homeworkExercise = models.ManyToManyField(Exercise, through = 'AssignHomework')
91     #uniquement les devoirs quiz
92     homeworkCourse = models.ManyToManyField(Course, through = 'AssignHomework')
93     #uniquement les devoirs cours
94     homeworkQuiz = models.ManyToManyField(Quiz, through = 'AssignHomework')
95     created_on = models.DateTimeField(auto_now=True)
96
97     def __str__(self):
98         return "Classe {0}".format(self.name)
99
100
101 #Table intermédiaire pour affecter un membre à un groupe
102
103 class GroupMembers(models.Model):
104     teacher = models.ForeignKey(Teacher, null = True)
105     student = models.ForeignKey(Student, null = True)
106     group = models.ForeignKey(Group)
107     added_on = models.DateTimeField(auto_now=True)
108
109 #Table intermédiaire pour assigner un devoir à un groupe
110
111 class AssignHomework(models.Model):
112     group = models.ForeignKey(Group)
113     exercise = models.ForeignKey(Exercise, null = True)
114     quiz = models.ForeignKey(Quiz, null = True)
115     course = models.ForeignKey(Course, null = True)
116     assigned_on = models.DateTimeField(auto_now=True)
117     date = models.CharField(max_length = 20, default = "Demain")

```

## 7.3.2 views.py

```

1  from django.shortcuts import render, redirect
2  from django.contrib.auth import authenticate, login, logout
3  from dashboard.forms import NewGroupForm, NewStudentForm, NewTeacherForm, \
4      AddHomeworkForm, NewPasswordForm
5  from django.core.urlresolvers import reverse
6  from common.models import Group, Teacher, GroupMembers, Student, \
7      AssignHomework, Exercise, Quiz, Course
8  from django.contrib.auth.models import User
9  from django.http import HttpResponseRedirect
10
11  #Accueil du dashboard
12  def home(request):
13      voyelle = 'aeiouyàâäëèêëîïïîôöõöüüüüAEIOUY' #Pour déterminer si le template affiche De ou D'
14      user = Teacher.objects.get(user = request.user)
15      firstLetter = request.user.username[0] #Idem
16      return render(request, 'dashboard/templates/dashboard/index.html', locals())
17
18  #Exercices, quiz et cours
19  def exercises(request):
20      user = Teacher.objects.get(user = request.user)
21      return render(request, 'dashboard/templates/dashboard/exercises.html', locals())
22
23  #Création de groupe
24  def newgroup(request):
25      success = False
26      user = Teacher.objects.get(user = request.user)
27      if request.method == "POST":
28          form = NewGroupForm(request.POST)
29          if form.is_valid():
30              group_name = form.cleaned_data["group_name"]
31
32              newGroup = Group.objects.create(name = group_name)
33              newGroup.save()
34              #Lie le Teacher et le groupe à travers la table intermédiaire
35              teacherToGroup = GroupMembers(teacher = user, group = newGroup)
36              teacherToGroup.save()
37              success = True #Pour retourner le message de confirmation
38      else:
39          form = NewGroupForm()
40      return render(request, "dashboard/templates/dashboard/newclass.html", locals())
41
42  #Changement de mot de passe
43  def profil(request):
44      user = Teacher.objects.get(user = request.user)
45      success = ''
46      if request.method == "POST":
47          userProfile = request.user
48          form = NewPasswordForm(request.POST)
49          if form.is_valid():
50              password = form.cleaned_data["password"]
51              passwordConfirm = form.cleaned_data["passwordConfirm"]
52              if password != passwordConfirm:
53                  success = False #Message d'erreur
54              else:
55                  success = True #Message de confirmation
56                  u = request.user
57                  u.set_password(password)
58                  u.save()
59      else:
60          form = NewPasswordForm()
61      return render(request, 'dashboard/templates/dashboard/profile.html', locals())

```

```

62
63 def group(request, group_id):
64     user = Teacher.objects.get(user = request.user)
65     group = Group.objects.get(id = group_id)
66
67     studentList = group.student.all()
68     teacherList = group.teacher.all()
69     homeworkExList = group.homeworkExercise.all() #
70     homeworkQuList = group.homeworkQuiz.all()    # Pour avoir la liste des devoirs
71     homeworkCoList = group.homeworkCourse.all()   # selon les genres d'activités
72
73     deleteConfirmation = False #Pour supprimer une classe
74
75     if request.method == "POST":
76
77         #Ajouter un professeur au groupe
78         if 'addTeacher' in request.POST:
79             erreurTeacher = False
80             formTeacher = NewTeacherForm(request.POST)
81             if formTeacher.is_valid():
82                 newTeacher = formTeacher.cleaned_data["nickname"]
83                 try:
84                     try:
85                         teacherUser = User.objects.get(username = newTeacher)
86                         teacher = Teacher.objects.get(user = teacherUser)
87                         newTeacherToGroup = GroupMembers(teacher = teacher, group = group)
88                         newTeacherToGroup.save()
89                     except User.DoesNotExist:
90                         erreurTeacher = True #Message d'erreur
91                 except Teacher.DoesNotExist:
92                     erreurTeacher = True #Idem
93
94         #Ajouter un élève au groupe
95         elif 'addStudent' in request.POST:
96             formStudent = NewStudentForm(request.POST)
97             erreurStudent = False
98             if formStudent.is_valid():
99                 try:
100                     try:
101                         newStudent = formStudent.cleaned_data["nickname"]
102                         studentUser = User.objects.get(username = newStudent)
103                         student = Student.objects.get(user = studentUser)
104                         newStudentToGroup = GroupMembers(student = student, group = group)
105                         newStudentToGroup.save()
106                     except User.DoesNotExist:
107                         erreurStudent = True #Message d'erreur
108                 except Student.DoesNotExist:
109                     erreurStudent = True #Idem
110
111         #Assigner un devoir
112         elif 'assignHomework' in request.POST:
113             formHomework = AddHomeworkForm(request.POST)
114             erreur = False
115             if formHomework.is_valid():
116                 homeworkid = formHomework.cleaned_data["homeworkid"]
117                 genre = formHomework.cleaned_data["genre"]
118
119                 #Cherche l'activité selon le genre choisi
120                 if genre == "exercice":
121                     try:
122                         exercise = Exercise.objects.get(id = homeworkid)
123                         newHomework = AssignHomework(exercise = exercise, group = group)
124

```

```

125         newHomework.save()
126     except Exercise.DoesNotExist:
127         erreur = True #Message d'erreur
128
129     if genre == "quiz":
130         try:
131             quiz = Quiz.objects.get(id = homeworkid)
132             newHomework = AssignHomework(quiz = quiz, group = group)
133             newHomework.save()
134         except Quiz.DoesNotExist:
135             erreur = True #Idem
136
137     if genre == "course":
138         try:
139             cours = Course.objects.get(id = homeworkid)
140             newHomework = AssignHomework(course = cours, group = group)
141             newHomework.save()
142         except Course.DoesNotExist:
143             erreur = True #Idem
144     elif 'deleteClass' in request.POST:
145         deleteConfirmation = True #Fait apparaître le deuxième bouton de confirmation
146
147     #Supprime la classe
148     elif 'deleteClassConfirm' in request.POST:
149         group = Group.objects.get(id = group_id)
150         group.delete()
151         return redirect('home')
152
153
154     formStudent = NewStudentForm()
155     formTeacher = NewTeacherForm()
156     formHomework = AddHomeworkForm()
157
158     else:
159         formStudent = NewStudentForm()
160         formTeacher = NewTeacherForm()
161         formHomework = AddHomeworkForm()
162     return render(request, 'dashboard/templates/dashboard/classe.html', locals())
163
164 #Retirer d'un groupe
165 def deleteFromGroup(request, member_id, group_id):
166     if request.method == "POST":
167
168         #Selon élève ou professeur
169         if 'deleteStudent' in request.POST:
170
171             student = Student.objects.get(id = member_id)
172             group = Group.objects.get(id = group_id)
173             studentToGroup = GroupMembers.objects.get(student = student, group = group)
174             studentToGroup.delete()
175
176
177         elif 'deleteTeacher' in request.POST:
178             teacher = Teacher.objects.get(id = member_id)
179             group = Group.objects.get(id = group_id)
180             teacherToGroup = GroupMembers.objects.get(teacher = teacher, group = group)
181             teacherToGroup.delete()
182
183     return redirect('group_view', group_id = group_id)
184
185 #Supprimer une activité
186 def deleteActivity(request, activity_id):
187     if request.method == "POST":

```

```

188
189     #Selon exercice, quiz ou cours
190     if 'deleteExercice' in request.POST:
191         exercise = Exercise.objects.get(id = activity_id)
192         exercise.delete()
193     if 'deleteQuiz' in request.POST:
194         quiz = Quiz.objects.get(id = activity_id)
195         quiz.delete()
196     if 'deleteCourse' in request.POST:
197         course = Course.objects.get(id = activity_id)
198         course.delete()
199     return redirect('exercices')
200
201 #Retirer un devoir
202 def deleteHomework(request, group_id, homework_id):
203     if request.method == "POST":
204
205         #Selon exercice, quiz ou cours
206         if 'deleteHomeworkEx' in request.POST:
207             exercise = Exercise.objects.filter(id = homework_id)
208             exercise = exercise[0]
209             group = Group.objects.get(id = group_id)
210             assignedHomework = AssignHomework.objects.filter(group = group, exercise = exercise)
211             assignedHomework = assignedHomework[0]
212             assignedHomework.delete()
213         if 'deleteHomeworkQu' in request.POST:
214             quiz = Quiz.objects.filter(id = homework_id)
215             quiz = quiz[0]
216             group = Group.objects.get(id = group_id)
217             assignedHomework = AssignHomework.objects.filter(group = group, quiz = quiz)
218             assignedHomework = assignedHomework[0]
219             assignedHomework.delete()
220         if 'deleteHomeworkCo' in request.POST:
221             course = Course.objects.filter(id = homework_id)
222             course = course[0]
223             group = Group.objects.get(id = group_id)
224             assignedHomework = AssignHomework.objects.filter(group = group, course = course)
225             assignedHomework = assignedHomework[0]
226             assignedHomework.delete()
227     return redirect('group_view', group_id = group_id)

```

### 7.3.3 urls.py

```

1 from django.conf.urls import patterns, include, url
2 from django.contrib import admin
3 from dashboard.views import *
4
5
6 urlpatterns = patterns('teachers.views',
7     url(r'^home/$', home, name='home'),
8     url(r'^classe/(?P<group_id>\d+)/$', group, name='group_view'),
9     url(r'^exercices/$', exercises, name='exercices'),
10    url(r'^nouveau_groupe/$', newgroup, name='newgroup'),
11    url(r'^profil/$', profil, name = 'profil'),
12    #Pour retirer d'un groupe
13    url(r'^enlever_groupe/(?P<group_id>\d+)/ (?P<member_id>\d+)/$',
14        deleteFromGroup, name = "deleteFromGroup"),
15    #Pour supprimer une activité
16    url(r'^enlever_activité/(?P<activity_id>\d+)/$',
17        deleteActivity, name = "deleteActivity"),
18    #Pour retirer un devoir

```



```
19         url(r'^enlever_devoir/(?P<group_id>\d+)/(?P<homework_id>\d+)/$',
20             deleteHomework, name = "deleteHomework"),
21     )
```

---

## Table des illustrations

---



---

### Table des illustrations

---

2.1	Formulaire de création de groupe . . . . .	5
2.2	Page d'administration d'un groupe . . . . .	6
2.3	Ce à quoi ressemble la page une fois que des membres ont été rajoutés . . . . .	6
2.4	Message d'erreur retourné si l'utilisateur n'est pas valable . . . . .	6
2.5	Différents champs à compléter pour assigner un devoir . . . . .	7
2.6	Message d'erreur retourné si l'activité n'a pas pu être trouvée . . . . .	7
2.7	Ce à quoi ressemble la page Exercices . . . . .	8
2.8	Apparence du menu déroulant . . . . .	8
2.9	Message pour confirmer que le changement de mot de passe a correctement eu lieu . . . . .	8
2.10	Message d'erreur retourné si les champs n'ont pas correctement été remplis . . . . .	9
3.1	Créer un workspace de type Custom . . . . .	11
3.2	Schéma de navigation du site . . . . .	12
3.3	Schéma résumant les actions qui se déroulent selon les utilisations du professeur . . . . .	13
3.4	Exemples de <code>glyphicons</code> dans le menu de gauche : l'oeil ou la maison . . . . .	13
3.5	Schéma résumant les relations entre les différents modèles . . . . .	15
4.1	Schéma résumant les 4 étapes du développement dirigé par les tests . . . . .	18