Rallye Lecture : Comptes utilisateurs Enseignants et authentification

Techniques mises en œuvre :

- ✓ Utilisation du Framework codeIgniter
- ✓ Utilisation du design pattern MVC
- ✓ Persistance des données au travers de classes DAO
- ✓ Envoi d'email
- ✓ Gestion et paramétrage d'une ReCaptcha

Objectif : faire évoluer la gestion des comptes utilisateurs en ajoutant les fonctionnalités suivantes :

- Inscription d'un nouveau professeur sur le site (register),
- L'inscription utilisera la ReCaptcha google pour vérifier que l'utilisateur n'est pas un robot,
- Activation du compte par email,
- Mot de passe oublié et permettre à l'utilisateur d'en changer (forgot password),
- Modification des informations de son propre compte.

Pour faciliter la mise en place d'un système d'authentification nous allons utiliser une bibliothèque. Une bibliothèque est une ou un ensemble de classes que nous allons pouvoir utiliser dans un développement voir même réutiliser ses dans de futurs développements.
Nous avons vu que CodeIgniter possède déjà des bibliothèques qui sont très bien présentées dans la documentation en ligne de CI (voir répertoires user_guide installé lors de chaque création de projet CI).
Pour rappel voici celles que nous avons déjà utilisées :

- ✓ File Uploading class,
- ✓ Form Validation class,
- ✓ Image Manipulation class,
- ✓ Session Class,
- ✓ Pagination Class,
- ✓ Database Class.

En plus de ces bibliothèques présentes lors de la création d'un projet CI, vous avez la possibilité de déployer vos propres librairies ou d'installer des librairies développées par des tiers. Concernant l'authentification il en existe un certain nombre :

- Ion Auth 2,
- Tank-Auth
- Oauth etc..

La bibliothèque que nous allons utiliser se nomme CodeIgnither-Aauth vous la trouverez sur github :

https://github.com/emreakay/CodeIgniter-Aauth

Ainsi que des exemples de code du wiki :

https://github.com/emreakay/CodeIgniter-Aauth/wiki/_pages

Elle est déjà en place dans le projet. Vous avez déjà :

- ✓ Installé le code dans : rallyeLecture/application/libraries/Aauth.php qui contient tout le code source et une classe nommée Aauth.
- ✓ Créé les tables nécessaires à la création et à la gestion des utilisateurs
- ✓ Créé le fichier de configuration de cette librairie dans : rallyeLecture/application/config/aauth.php

C'est à peu près toujours les trois étapes que l'on doit respecter lorsque l'on configure l'installation d'une bibliothèque provenant d'un développement effectué par une tierce personne.

Je vous donne aussi un lien github où sont référencées tout un tas de librairies CI que l'on peut utiliser :

https://github.com/codeigniter-id/awesome-codeigniter

Certaines sont très bien documentées et leur code et robuste et équivalent à un développement professionnel d'autres sont de simples boites à outils moins bien documentées, mais toutes aussi utiles (il faut parfois se plonger dans leur code pour en comprendre leur fonctionnement).

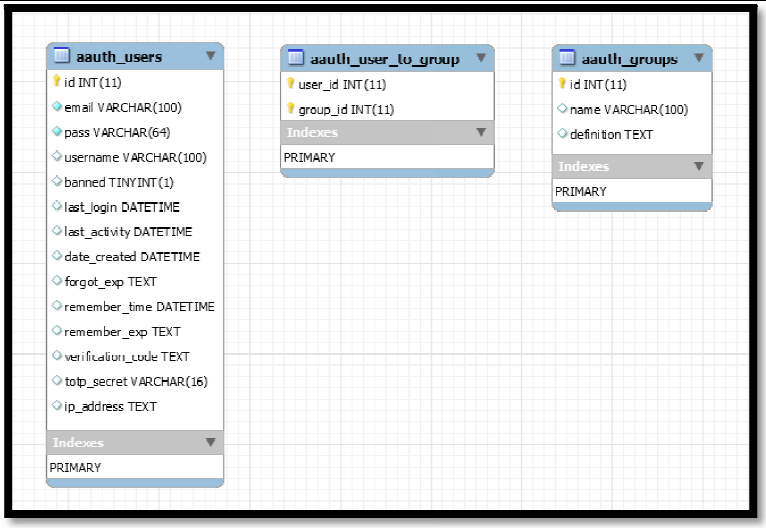
Classe Aauth

Aauth

CI : object
config_vars : array
errors : array
infos : array
flash_errors : mixed
flash_infos : mixed
aauth_db : object

__construct()
login(identifier : mixed,pass : string,remember : bool,totp_code : mixed)
is_loggedin()
control(perm_par : bool)
logout()
login_fast(user_id : int)
reset_login_attempts()
remind_password(email : string)
reset_password(ver_code : string)
update_last_login(user_id : int|bool)
update_login_attempts()
get_login_attempts()
update_remember(user_id : int,expression : int,expire : int)
create_user(email : string,pass : string,username : string)
update_user(user_id : int,email : string|bool,pass : string|bool,username : mixed)
list_users(group_par : bool|int,limit : string,offset : bool,include_banneds : bool)
get_user(user_id : int|bool)
verify_user(user_id : int,ver_code : string)
send_verification(user_id : int)
delete_user(user_id : int)
ban_user(user_id : int)
unban_user(user_id : int)
is_banned(user_id : int)
user_exist_by_username(name : mixed)
user_exist_by_name(name : mixed)
user_exist_by_email(user_email : *)
user_exist_by_id(user_id : mixed)
get_user_id(email : string|bool)
get_user_groups(user_id : int|bool)
update_activity(user_id : int|bool)
hash_password(pass : string,userid : *)
verify_password(password : string,hash : string)
create_group(group_name : string,definition : string)
update_group(group_par : mixed,group_name : string,definition : mixed)
delete_group(group_par : mixed)
add_member(user_id : int,group_par : int|string)
remove_member(user_id : int,group_par : int|string)
add_subgroup(group_par : int|string,subgroup_par : mixed)
remove_subgroup(group_par : int|string,subgroup_par : int|string)
remove_member_from_all(user_id : int)
is_member(group_par : int|string,user_id : int|bool)
is_admin(user_id : int)
list_groups()
get_group_name(group_id : int)
get_group_id(group_par : int|string)
get_subgroups(group_par : int|string)
create_perm(perm_name : string,definition : string)
update_perm(perm_par : int|string,perm_name : string,definition : string)
delete_perm(perm_par : int|string)
is_allowed(perm_par : int,user_id : int|bool)
is_group_allowed(perm_par : int,group_par : int|string|bool)
allow_user(user_id : int,perm_par : int)
deny_user(user_id : int,perm_par : int)
allow_group(group_par : int|string|bool,perm_par : int)
deny_group(group_par : int|string|bool,perm_par : int)
list_perms()
get_perm_id(perm_par : int|string)
send_pm(sender_id : int,receiver_id : int,title : string,message : string)
send_pms(sender_id : int,receiver_ids : array,title : string,message : string)
list_pms(limit : int,offset : int,receiver_id : int,sender_id : int)
get_pm(pm_id : int,user_id : int,set_as_read : bool)
delete_pm(pm_id : int,user_id : mixed)
cleanup_pms()
count_unread_pms(receiver_id : int|bool)
set_as_read_pm(pm_id : int)
error(message : string,flashdata : boolean)
keep_errors(include_non_flash : boolean)
get_errors_array()
print_errors(divider : string)
clear_errors()
info(message : string,flashdata : boolean)
keep_infos(include_non_flash : boolean)
get_infos_array()
print_infos(divider : string)
clear_infos()
set_user_var(key : string,value : string,user_id : int)
unset_user_var(key : string,user_id : int)
get_user_var(key : string,user_id : int)
get_user_vars(user_id : int)
list_user_var_keys(user_id : int)
generate_recaptcha_field()
update_user_totp_secret(user_id : mixed,secret : mixed)
generate_unique_totp_secret()
generate_totp_qrcode(secret : mixed)
verify_user_totp_code(totp_code : mixed,user_id : mixed)
is_totp_required()

Tables liées à la gestion des utilisateurs



Pré requis 1 correction de bug dans la classe application/libraries/Aauth.php

Apparemment au fur et à mesure de l'utilisation de la bibliothèque Aauth il me semble que deux bugs sont présents dans la classe et empêchent de l'utiliser convenablement pour réaliser nos différents travaux :

- oubli d'un paramètre dans la méthode is_admin(). Cette méthode permet de tester qu'un utilisateur est ou n'est pas un administrateur. Problème l'id de l'utilisateur Aauth a été oublié. Donc impossible de tester ce cas. "ligne 704" dans la méthode create_user du source, ajouter le paramètre :

```
if($this->config_vars['verification'] && !$this->is_admin($user_id)){
```

- Autre bug lorsque la recaptcha est testé on a le droit à un certain nombre d'essais. La comparaison a été inversée "ligne 2428" du code source.

```
if($this->config_vars['ddos_protection'] && $this->config_vars['recaptcha_active'] &&
        $this - >get_login_attempts() <= $this->config_vars['recaptcha_login_attempts']){
```

Pré requis 2 correction de bug dans la vue EnseignantIndex.php

```
<!-- navigation -->
<div class="navigation">
    <a href="<?php echo base_url(); ?>">Home</a>
    <a href="<?php echo base_url('Enseignant/Add'); ?>">Ajouter</a>
</div>
```

Le href vers Enseignant/Add doit être modifié tel quel.

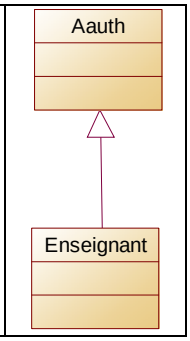
Pré requis 3 correction de bug dans le contrôleur Enseignant.php

```
84
85      function remove($id) {
86          $enseignant=$this->enseignantModel->get_enseignant($id);
87          if (isset($enseignant['id'])) {
88              $this->aauth->delete_user($enseignant['idAuth']);
89              $this->enseignantModel->delete_enseignant($id);
90              redirect('Enseignant/Index');
91          }
92          else {
93              show_error("L'enseignant que vous essayez de supprimer n'existe pas");
94          }
95      }
```

Lorsqu'un enseignant est supprimé on doit supprimer son compte de connexion.

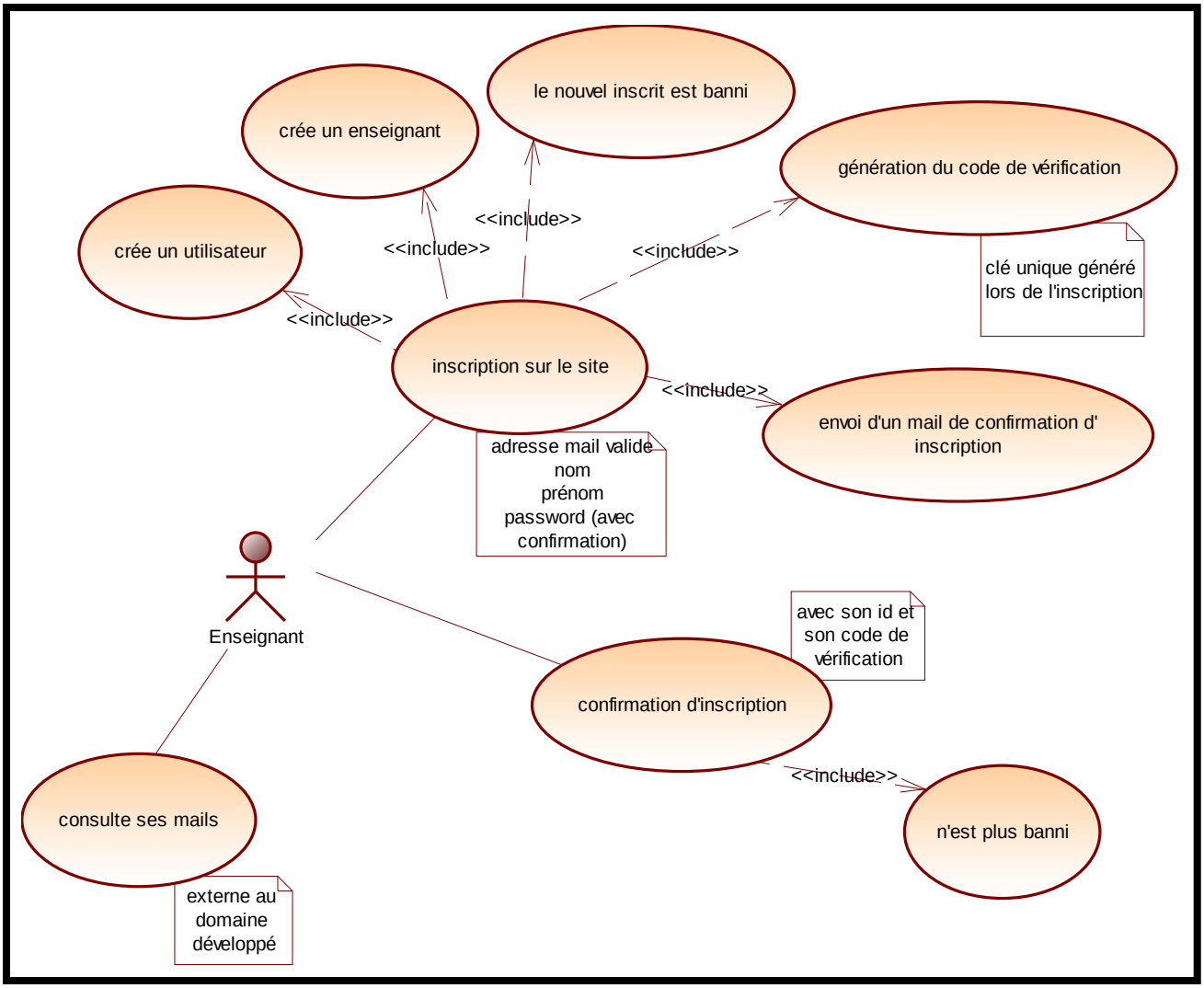
En fait un enseignant ne peut pas exister sans qu'il possède un compte Aauth. On est en train de gérer un lien d'héritage entre deux tables de base de données. Cela a des répercussions sur toutes les actions en base de données (insert, update, delete). Lorsque chacune de ces opérations est effectuée, il faut penser à la répercussion sur la table Aauth :

- Insert Enseignant => insert dans Aauth et récupération de l'id de Aauth généré.
- Update Enseignant => aucune répercussion sur la table Aauth.
- Delete Enseignant => delete de l'occurrence correspondante dans Aauth (grace à l'id de Aauth).



Inscription d'un enseignant

Scénario d'inscription d'un enseignant sur le site :



Description textuel du cas d'utilisation.

1. l'enseignant demande à s'inscrire
2. Il saisit :
    a. son nom
    b. son prénom
    c. une adresse mail valide
    d. un mot de passe
    e. il confirme son mot de passe
3. Il crée son compte
4. Le système crée un enseignant dans la table enseignant
5. Le système crée un utilisateur dans la table aauth
6. Cet utilisateur est momentanément banni (il ne peut se connecter au site) tant qu'il n'a pas confirmé son inscription
7. Un mail de confirmation est transmis à l'enseignant sur sa boite mail
8. L'enseignant click sur le lien du message mail de confirmation
9. Le système vérifie le code confirmation, le compte n'est plus banni.
10. L'utilisateur peut se connecter sur le site.

1. Modification du paramétrage de la bibliothèque à Auth et CI.

La classe à Aauth prévoit déjà ce genre de scénario : nous devons simplement activer l'envoi du mail lors de la création d'un nouvel utilisateur (méthode create_user de la classe Aauth).

Nous modifions le fichier de paramétrage : application/config/aauth.php :

`'verification' est passé à true`

```
128
129     'login_with_name'               => false,
130
131     'email'                         => 'admin@sio.jjr.fr',
132     'name'                          => 'Admin',
133     'email_config'                  => false,
134
        'verification'                  => true,
136     'verification_link'             => '/account/verification/',
137     'reset_password_link'           => '/account/reset_password/',
138
139     'hash'                          => 'sha256',
140     'use_password_hash'             => false,
141     'password_hash_algo'            => PASSWORD_DEFAULT,
142     'password_hash_options'         => array(),
143
144     'pm_encryption'                 => false,
145     'pm_cleanup_max_age'            => "3 months",
146   );
147
148   $config['aauth'] = $config_aauth['default'];
149
150 /* End of file aauth.php */
151 /* Location: ./application/config/aauth.php */
```

Nous ne le modifions pas mais vous pouvez voir le paramètre 'verification_link' qui est le lien que doit rappeler notre enseignant sur le site pour valider son inscription.

L'envoi d'email est une fonctionnalité liée à CI nous devons le paramétrer :

Nous modifions le paramétrage de application/config/autoload.php :

```
93  /*
94  | -------------------------------------------------------------------
95  |  Auto-load Config files
96  | -------------------------------------------------------------------
97  | Prototype:
98  |
99  |       $autoload['config'] = array('config1', 'config2');
100 |
101 | NOTE: This item is intended for use ONLY if you have created custom
102 | config files.  Otherwise, leave it blank.
103 |
104 */
    $autoload['config'] = array('email');
106
```

Nous créons dans application/config/email.php un nouveau fichier de configuration qui contient tous les paramètres spécifiques d'envoi d'un email.

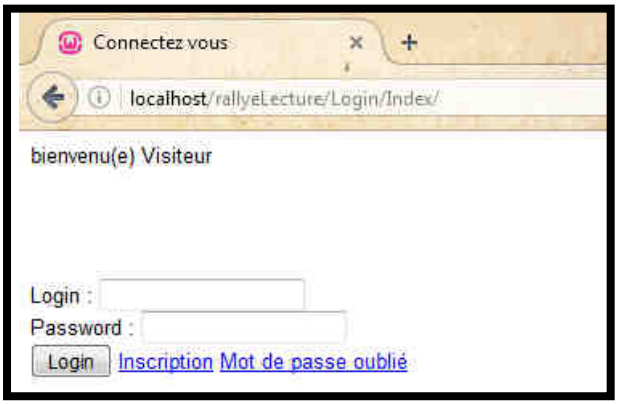
```php
<?php

$config['protocol']='smtp';
$config['smtp_host']='ssl://smtp.gmail.com';
$config['smtp_port']='465';
$config['smtp_timeout']='7';
$config['smtp_user']='ppe.slam.sio.jjr@gmail.com';
$config['smtp_pass']='siojjrsiojjr';
$config['charset']='utf-8';
$config['newline']="\r\n";
$config['mailtype'] = 'text';
```

Ceci est un exemple merci de créer un compte gmail par groupe de développement au format suivant, c'est important car ce compte gmail nous resservira pour la recaptcha :
nomDuGroupe.ppe.slam.sio.jjr@gmail.com Utilisez le mot de passe siojjrsiojjr pour ce compte.

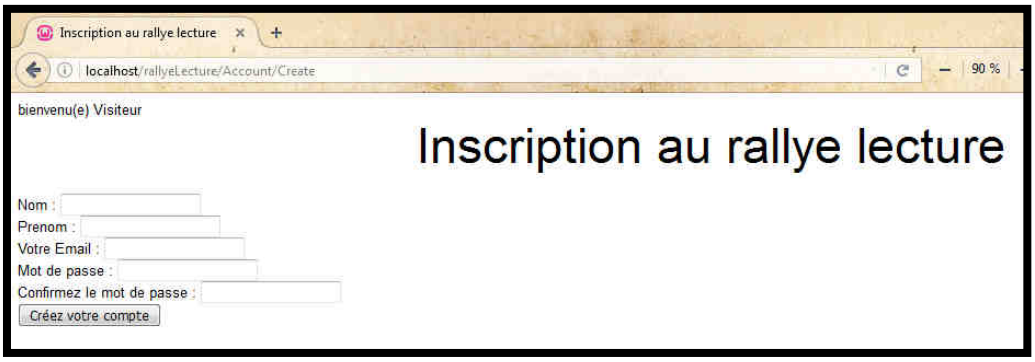
2. Modification de la vue Login.php

On ajoute deux liens l'un permettant la création d'un nouveau compte, l'autre qui nous servira plus tard et permettant de demander la modification d'un mot de passe oublié.



3. Création des vues de la class Account

Créez la vue AccountCreate.php



Créez les vues :

AccountConfirmation.php dont voici le code

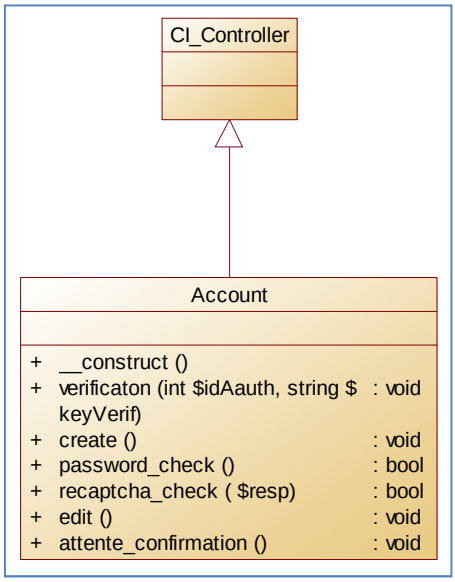
```
Merci de vous être inscrit sur notre site.<br>
Votre code de confirmation d'inscription vous a été transmis sur votre boite mail :
<?php echo $email;?> <br>
Cliquez sur le lien que contient le message et votre inscription sera complète.<br>
Vous pourrez ensuite vous connecter sur votre espace rallye lecteur.
```

et la vue AccountInscrit.php

```
Votre procédure d'inscription est terminée<br>
Vous pouvez maintenant vous connecter : <a href="<?php echo base_url('Login/Index'); ?>">Login</a>
```

4. Mise en place du Contrôleur Account

Voici le diagramme de classe du contrôleur Account, Implémentez le squelette de cette classe.



Dans le constructeur pensez à :

- Définir le modèle de la classe d'enseignant,
- Charger la bibliothèque de la classe Aauth,
- Charger la bibliothèque de form_validation.

Méthode create :

Première chose on charge les règles de validation liées à l'enseignant qui seront contrôlées par le formulaire. Ces règles sont stockées dans le modèle et sont chargées grace à la fonction LoadValidationRules.

```
LoadValidationRules($this->enseignantModel,$this->form_validation);
```

Les trois règles de validation sont chargées d'un seul coup

```
5   */
6   class EnseignantModel extends CI_Model {
7       public $ValidationRules=array(
8           array('field'=>'nom','label'=>'Nom','rules'=>'required|max_length[45]'),
9           array('field'=>'prenom','label'=>'Prenom','rules'=>'required|max_length[45]'),
10          array('field'=>'login','label'=>'Login','rules'=>'required|max_length[100]|valid_email')
11      );
12
```

Nous allons ajouter maintenant les règles de validation spécifiques à la saisie du mot de passe et à la confirmation du mot de passe. En effet ces règles sont spécifiques à la création de l'occurrence dans la table Aauth.

pour cela on ajoute :

```
$this->form_validation->set_rules('password','Password','required|max_length[100]');
$this->form_validation->set_rules('passwordConfirmation',
                        'Confirmez le mot de passe',"required|max_length[100]|callback_password_check");
```

Vous noterez aussi que la règle liée à la confirmation du mot de passe n'est pas une règle standard. Il va falloir la coder dans une méthode spécifique (password_check) que nous allons ajouter au contrôleur. CI demande à ce que les règles spécifique soient préfixées du mot "calllback_".

Comportement e CI avec une règle de validation spécifique : lorsque CI détecte une telle règle, il appelle la méthode spécifiée. Si celle-ci retourne vrai il considère que la règle est validée et le traitement suit son cours. Dans le cas contraire il charge et affiche le message d'erreur que l'on lui spécifie.

Nous allons donc ajouter la méthode suivante :

```
public function password_check() {
     $password=$this->input->post('password');
     $passwordConfirmation=$this->input->post('passwordConfirmation');
     if ($password!=$passwordConfirmation) {
         $this->form_validation->set_message('password_check',
                         'le mot de passe de confirmation est différent du mot de passe initial');
         return false;
     }
     else {
         return true;
     }
}
```

Il vous reste à coder la partie form_validation :

⇒ En cas de succès on :

- ✓ crée le nouvel utilisateur dans la table aauth (utilisation de la méthode aauth - > create_user($email,$password).
- ✓ récupère le nom, le prénom, l'id de l'occurrence de la table Aauth que l'on vient de créer et on crée l'enseignant. Méthode add_Enseignant de la classe EnseignantModel.
- ✓ on ajoute le nouvel enseignant créé au groupe "Enseignant" (utilisation de la méthode aauth->add_member
- ✓ on appelle la méthode attente_confirmation pour afficher à l'utilisateur que l'on attend sa confirmation.

⇒ En cas d'erreur ou lors du premier affichage :

- ✓ On charge la vue "AccountCreate" avec le titre "Inscription au rallye lecture".
- ✓ Ne pas oublier d'afficher l'entête et le pied de page.

Code de la méthode attente_confirmation :

```
   public function attente_confirmation($email) {
        $data['title']="Confirmation de votre inscription";
        $data['email']=$email;
        $this->load->view('AppHeader',$data);
        $this->load->view('AccountConfirmation',$data);
        $this->load->view('AppFooter');
    }
```

Enfin si tout c'est bien passé à l'issu de l'inscription d'un utilisateur, celui-ci devrait reçevoir dans boite un message proche de ceci :

Si vous avez bien suivi jusqu'à maintenant vous devriez comprendre à quoi correspond ce lien sur votre site et savoir quelle méthode coder.

Il ne vous reste plus qu'à coder la méthode verification($idAuth, $keyValue) qui doit :

⇒ faire appel à la méthode aauth->verify_user qui vérifie que l'utilisateur existe, que le code de confirmation est bien celui attendu (le code de verification est stocke dans la table aauth vous pouvez vérifier que le code stocké correspond bien au code du mail) et "débanni" l'utilisateur pour qu'il puisse se connecter.

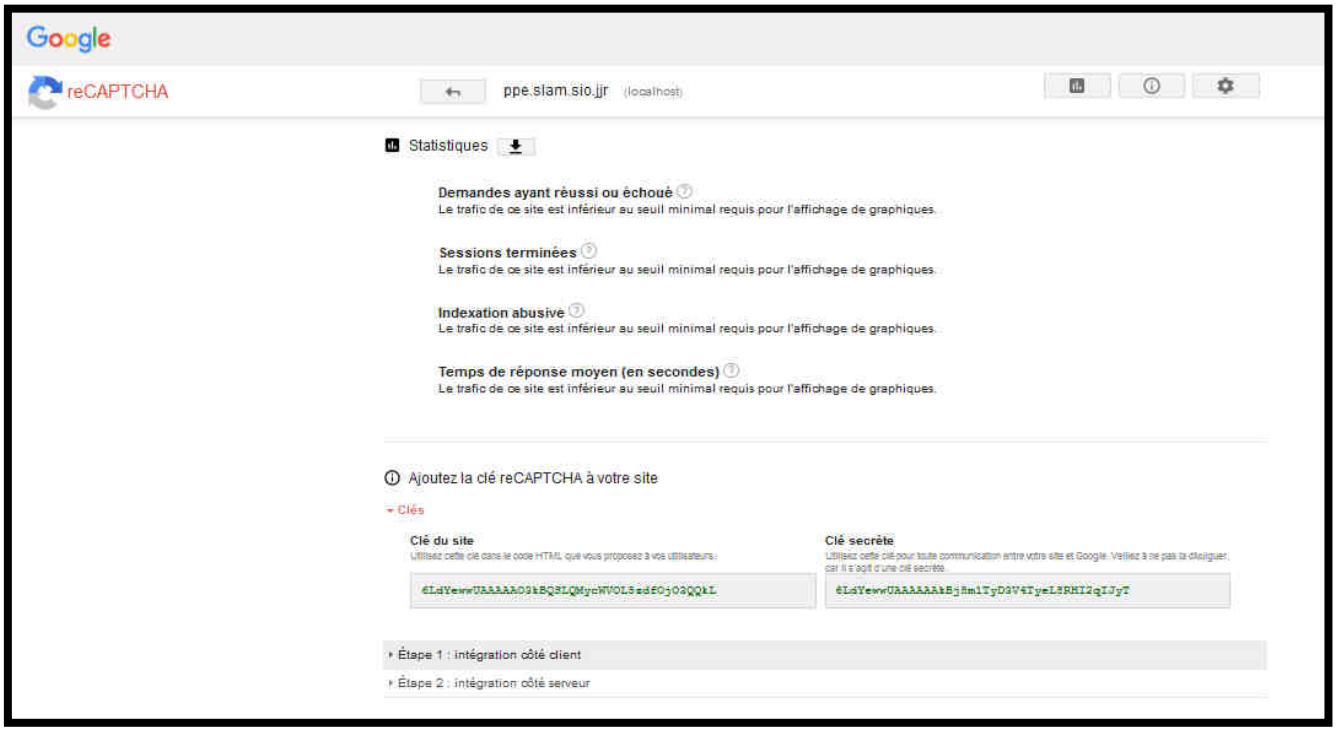
⇒ La confirmation doit se terminer pat l'affichage de la vue "AccountInscrit" que vous avez codé plus haut.

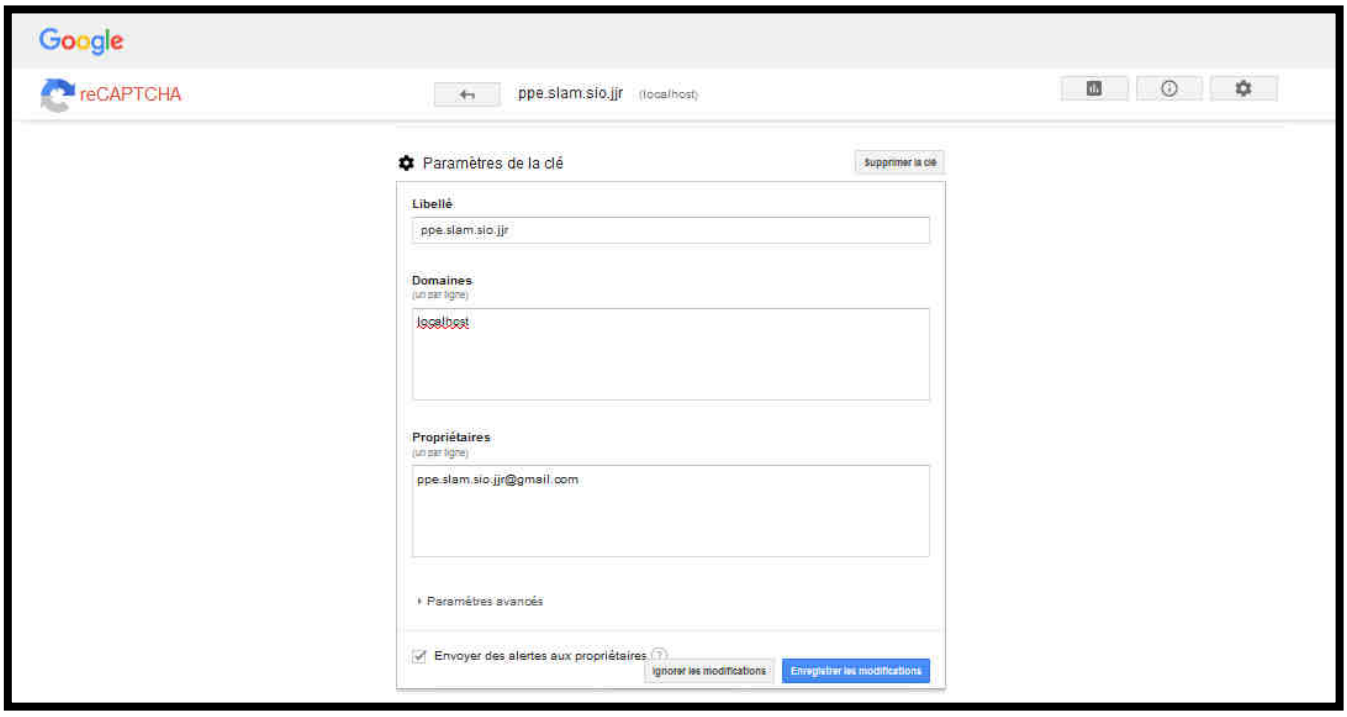
5. Etes vous un robot ? : Ajout de recaptcha dans votre application.

recaptcha est un service proposé par google qui permet de vérifier qu'une personne qui se connecte ou effectue n'importe quelle opération sur votre site n'est pas un robot. Dans le cas contraire un utilisateur mal intentionné risquerait de créer un script permettant par exemple de multiplier les inscriptions sur votre site et ainsi saturer votre base de données et générer un DoS sur votre site.

Pré requis 1 : paramétrage de recaptcha chez google.

Connectez-vous sur le compte gmail que vous avez créé plus haut (votreGroupe.ppe.slam.sio.jjr@gmail.com).
allez sur le lien suivant qui permet de paramétrer recaptcha : https://www.google.com/recaptcha/admin





Les trois paramètres importants sont les 2 clés, et le domaine : localhost pour pouvoir tester.

Pré requis 2 : paramétrage recaptcha dans CI.

Paramètres de la recaptcha dans application/config/aauth.php

```
113
114     'recaptcha_active'              => true,
115     'recaptcha_login_attempts'      => 4,
116     'recaptcha_siteKey'             => '6LdYewwUAAAAAO3kBQSLQMycWVOL5zdfOjO3QQkL',
117     'recaptcha_secret'              => '6LdYewwUAAAAAAkBj8m1TyD3V4TyeL8RHI2qIJyT',
118
```

Les clés publiques et privées sont issues du site de google. La clé privée ne doit pas être dévoilée.

Modificiation de la vue

Modification Vue AccountCreate.php

```
<?php echo $this->aauth->generate_recaptcha_field(); ?>
<button type="submit">Créez votre compte</button>
<?php echo form_close(); ?>
```

Modification du contrôleur de Account

✓ Ajout d'une règle de validation liée à la captcha

```
        $this->form_validation->set_rules('g-recaptcha-response','Captcha','callback_recaptcha_check');
```

✓ Ajout d'une méthode de validation controlant que la captcha a bien été validée

```
    public function recaptcha_check($resp) {
        if (empty($resp)) {
            $this->form_validation->set_message('recaptcha_check',
                        'quelque chose me dit que vous êtes un robot. Voulez vous essayer à nouveau ?');
            return false;
        }
        else {
            return true;
        }
    }
```

A vous

Travail à faire

1) Faites évoluer le menu dans application/models/menu.php

Ajoutez une option de menu ('Account','Edit','Mon Compte')

```
39              case 'Enseignant':
41                  return array(
42                      array('EnseignantClasse','Index','Mes Classes'),
43                      array('EnseignantLivre','Index','La Bibliothèque'),
44                      array('EnseignantRallye','Index','Les Rallyes'),
45                      array('TableauBord','Index','Tableau de Bord'),
46                      array('Account','Edit','Mon Compte'),
47                      array('Login','Index','Login'),
48                      array('Logout','Index','Logout')
49                  );
51              case 'Elève':
52                  return array(
53                      array('RallyeEleve','Index','Tes Rallyes'),
54                      array('Login','Index','Login'),
55                      array('Logout','Index','Logout')
57                  );
58              default: // Visiteur (la plus restrictive)
59                  return array(
60                      array('Login','Index','Login')
61                  );
62          }
      }
  }
```

Faites évoluer l'application pour qu'un utilisateur puisse changer les caractéristiques de son compte lorsqu'il est logué à l'application.

2) Perte de mot de passe
   a) Proposez un cas d'utilisation pertinent permettant à un utilisateur de prévenir le site qu'il a perdu son mot de passe, le site lui propose alors de renouveler et de confirmer son nouveau mot de passe.
   b) Identifiez les méthodes qui vous seraient utiles dans la classe aauth.
   c) proposez les différentes vues
   d) une fois l'ensemble validé (par moi) codez la solution.

Voici un exemple de scénario plausible pour la récupération d'un mot de passe :

Use case : mot de passe oublié.
1. Un utilisateur clique sur mot de passe oublié. Lien sur la page login/forgotPassword 2. L'utilisateur est dirigé vers une page de rappel account/rememberPassword où il entre son nom d'utilisateur. 3. Un e-mail contenant un lien vers une nouvelle page de changement de mot de passe est envoyé à l'utilisateur. 4. Ce lien reste valide pour une période de temps spécifiée. L'utilisateur réinitialise son mot de passe sur cette page et reçoit ensuite un courriel confirmant le changement du mot de passe.

Brouillons

Questionnement possible :
Gérer la désinscription d'une personne.
Ajouter table des adresses académiques, vérifier que le compte mail est bien académique lors de l'inscription.
L'enseignant souhaite une fonctionnalité permettant d'inscrire ses élèves à partir d'un fichier excel.