

# Dynamisation d'un site web

## Rapport de stage

LE PAGE Vincent  
Université de Sherbrooke  
faculté de génie

Tuteur de stage :  
M. Ruben Gonzalez-Rubio

Enseignant tuteur :  
Mme Anne-Isabelle Llanta





# Dynamisation d'un site web

## Rapport de stage

LE PAGE Vincent  
Université de Sherbrooke  
faculté de génie

Tuteur de stage :  
M. Ruben Gonzalez-Rubio

Enseignant tuteur :  
Mme Anne-Isabelle Llanta



## Remerciements

Je souhaite remercier toutes les personnes qui ont contribué au succès de mon stage.

Tout d'abord, j'adresse mes remerciements à mon professeur, **Mme Anne-Isabelle Llanta** qui m'a proposé ce stage et m'a permis de postuler dans celui-ci. De plus, elle a été présente pour répondre à mes questions à propos du Canada et des démarches administratives.

Je tiens à remercier vivement mon maître de stage, **M. Ruben Gonzalez-Rubio**, professeur titulaire à l'Université de Sherbrooke, pour son accueil, son accompagnement et ses conseils prodigués pendant ce stage.







# Sommaire :

<b>Introduction</b>	<b>5</b>
<b>1. La structure d'accueil</b>	<b>6</b>
1.1 Histoire	6
1.2 Fonctionnement de l'université	6
1.3 Environnement de travail	6
<b>2. Objectifs du projet</b>	<b>8</b>
2.1 Le sujet de stage	8
2.2 Les différentes tâches à réaliser	8
2.3 Sprint backlog	8
<b>3. Organisation du projet</b>	<b>11</b>
3.1 La méthode SCRUM	11
3.2 La gestion de notre projet	13
3.3 Un long apprentissage	14
<b>4. Réalisations techniques</b>	<b>15</b>
4.1 Mise en place de l'environnement	15
4.2 Le modèle MVC	15
4.3 Clean Code	16
4.4 Envoyer un mail	16
4.5 Une page type : la page profile	18
4.5.1 L'accès à la page	18
4.5.2 Scala et les vues	20
4.6 Les messages comme constante	21
4.7 Les tests	21
<b>Conclusion</b>	<b>24</b>
<b>Résumé</b>	<b>25</b>
<b>Abstract</b>	<b>27</b>
<b>Glossaire</b>	<b>29</b>
<b>Table des matières des annexes :</b>	<b>31</b>
1. Tableau caractérisation d'une organisation	32
2. Schéma model MVC	33



# Introduction

Ce stage a été réalisé dans le cadre de ma formation pour l'obtention du DUT Informatique à Lannion. Le stage a été effectué du 16 Avril 2018 au 27 Juin 2018 (11 semaines) à la faculté de génie de l'Université de Sherbrooke au Canada.

Tout d'abord réfractaire à l'idée de réaliser mon stage à l'étranger, c'est lorsque que ce stage en particulier a été proposé que je me suis questionné sur la possibilité d'effectuer un stage à l'étranger. En effet, l'idée de voyager et de découvrir un nouveau pays m'a beaucoup plu, d'autant plus que le Canada m'attirait beaucoup par son style de vie et sa culture. Un autre grand facteur de cette décision a été le sujet de stage qui correspondait parfaitement à ce que je souhaitais découvrir ou approfondir en informatique, c'est-à-dire le développement web. Alliant un sujet de stage et un lieu idéal je n'ai pas hésité très longtemps avant de postuler pour celui-ci.

Initialement, le stage portait sur la mise en place d'un site web de mise en ligne de CV pour les chercheurs de l'université. Nous verrons que le sujet a évolué durant la période de stage. Le professeur Gonzalez-Rubio nous a confié la tâche de réaliser un prototype de ce site web pour soumettre celui-ci à une commission, afin d'ensuite mettre en place une équipe de développement sur ce projet.

Pour commencer, je présenterai l'université de Sherbrooke ainsi que la place que j'y tenais. Par la suite, nous nous arrêterons sur les objectifs et besoins précis du projet. Puis j'expliquerai l'organisation mise en place afin de mener à bien ce projet. Enfin, nous verrons comment ce sujet a-t-il été réalisé, les réalisations techniques.

# 1. La structure d'accueil

## 1.1 Histoire

L'Université de Sherbrooke a vu le jour le quatre mai 1954 dans le but de répondre à un désir d'établir une université catholique francophone dans une région initialement à forte densité anglophone. En effet, jusqu'à cette date les seules universités francophones du Canada se situaient dans deux grandes villes, Montréal et Québec.

Avec un noyau initial de trois facultés - les arts, les sciences et le droits - l'Université se développe progressivement pour répondre aux besoins éducatifs de l'Estrie (Région administrative à l'Est de Montréal), et apporte une contribution originale en matière d'enseignement et de recherche. Au fil des années, l'Université a accueilli de nouvelles facultés comme le commerce, l'éducation, la médecine, le génie et l'éducation sportive. Elle accueille aujourd'hui plus de 40 000 étudiants.

## 1.2 Fonctionnement de l'université

Chacune des huit facultés de l'Université de Sherbrooke est dirigée par un doyen ou une doyenne, chacun assisté par un ou plusieurs vice-doyen ou vice-doyennes ainsi qu'un ou une secrétaire de faculté. Toutes les facultés sont sous la responsabilité du recteur ou rectrice.

## 1.3 Environnement de travail

Notre équipe était constitué de quatre étudiants, Loïc Travaillé et Julien Beuve, en DUT GEII, et Emilien Cosnier et moi-même en DUT Informatique. Nous étions supervisés par notre tuteur de stage M. Ruben Gonzalez-Rubio. L'université nous a

attribué à chacun un bureau dans une salle type open-space. Une chance pour nous car ce type de bureau a permis une communication facile et efficace entre les membres de l'équipe pendant notre stage. Nous avons également reçu un tableau et diverses fournitures pour organiser le projet. Les membres du projet étaient libres de choisir d'utiliser un ordinateur fournis par l'université ou un ordinateur personnel. J'ai choisi d'utiliser mon ordinateur personnel.

## 2. Objectifs du projet

### 2.1 Le sujet de stage

Le sujet de stage initial ayant été annulé, notre tuteur s'est donc penché sur un autre sujet qu'il souhaitait voir réalisé : la création d'un prototype de site web dynamique présentant les différents groupes de recherche et projets en cours. La nécessité de créer un prototype vient du fait que l'Université de Sherbrooke demande un prototype avant de financer un projet et de débloquer une équipe de développement.

### 2.2 Les différentes tâches à réaliser

Suivant la méthode SCRUM, chaque **sprint backlog** était créé à la fin du précédent **sprint**. Les grands objectifs de notre projet ont été d'instaurer un système de compte personnel sur le site, la gestion des équipes de projet, la navigation de projet en projet, et la gestion de compte. Nous avons également comme objectifs supplémentaires la mise en place de la possibilité d'afficher le site en anglais et en français ou l'affichage de photo avec leur modification disponible par l'utilisateur.

### 2.3 Sprint backlog

Dans cette partie je vais lister le contenu des différents *sprint* réalisés.

Sprint 1 (3 semaines) :

- Login (model, vue et contrôleur)
- Création d'une boîte mail
- New password (model, vue et contrôleur)

- Home (vue, contrôleur)
- Template (header, footer et menu)
- Charte graphique
- Configurer SVN
- Base de données (users et article)
- Test New password
- Template test
- Article test
- Login test
- Home test
- Users test
- Clean code
- Message d'erreur et ajouter lien dans login pour new password

#### Sprint 2 (2 semaines) :

- Home partie photo
- Edit home sans photo
- Edit home partie photo
- Page register
- Page profil
- Profil partie photo
- Edit profil sans photo
- EditProfil partie photo
- Edit team
- Team ancien membres
- Team active
- Gestion liste utilisateur
- Model
- Traducteur EN/FR

### Sprint 3 (2 semaines) :

- Team
- Gestion team
- past team
- Past team gestion
- Edit team
- Edit profile
- myProfile
- Edit myProfile
- Profile gestion
- Model EN/FR
- Template views à organiser
- Home gestion
- Edit home
- Project
- Add project
- Project gestion

### Sprint 4 (2 semaines) :

- Test Login
- Test Home
- Test Profil
- Test Register
- Test Team
- Photos Project et profil

## 3. Organisation du projet

### 3.1 La méthode SCRUM

Scrum signifie en anglais “mêlée” (référence au rugby), ce termes est utilisé pour souligner l’intense collaboration entre les différentes parties prenantes d’un projet. Scrum se particularise par la forte présence de divisions, dans le temps, dans les besoins et dans l’équipe.

Le temps se découpe en plusieurs blocs de temps appelées *itération* ou *sprint* très court ( entre deux et quatre semaines) en fonction du produit, de la taille d’équipe ou du contexte. A la fin d’un *sprint*, les équipes fournissent une partie du projet certe partielle mais fonctionnelle. A ce moment le client peut choisir de stopper le projet sans soucis puisqu’un produit partiel potentiellement livrable a été fournis. Il peut manquer certaine fonction au produit mais le plus important est là, il est fonctionnel.

Les besoins regroupés dans un **product backlog**, sont divisés en sous-besoins nommés *user-stories*. Ces *users-stories* suivent deux règles : elles doivent être estimables pour l’équipe en terme de temps, mais aussi pouvoir être réalisées le temps d’un seul et unique *sprint*. Elles sont ensuite ajoutées dans un *sprint* spécifique formant un **sprint backlog** (ensemble des tâches à réaliser pendant le *sprint*).

Enfin, diviser la taille d'une équipe revient à limiter la taille afin d'avoir la meilleure synergie.

Trois rôles sont définis par Scrum :

- Le **product owner** représente le client et la vision final du projet.
- Le **scrum-master** veille à que chaque membre de l'équipe puisse travailler à ces capacités maximales. Il protège aussi l'équipe des perturbations extérieures.
- L'équipe de développement celle mandatée pour réaliser le produit.

Le déroulement d'un projet scrum est jalonné de quatre réunions :

- Le **sprint planning** - comme son nom l'indique, c'est le moment où le contenu du *sprint* va être défini.
- Les **daily scrum** - chaque jour pendant un court temps, les membres de l'équipe se réunissent pour faire le point sur leur avancement. Chaque membre doit répondre à trois questions, qu'est ce que j'ai fait hier ? Qu'est ce que je vais faire aujourd'hui ? Est-ce que j'ai des obstacles qui m'empêche d'avancer ?
- Le **sprint review** - il s'agit d'une réunion bilan pendant laquelle le contenu du *sprint* va être présenté au client ou au *product owner*. C'est un moment crucial car il permet de récolter le **feedback** (point de vue du client sur ce qui a été réalisé) du client. Le client pourra donc réajuster ses besoins ou les ré-exprimer.
- La **rétrospective** - cette étape permet aux membres de l'équipe de discuter des forces et faiblesses mises au jour pendant le *sprint* précédent. Cette réunion est importante par sa capacité à empêcher la redondance d'un problème et à exploiter les forces.

Propre à toutes les méthodes agiles, un tableau peut être mis en place pour représenter l'avancé de projet. Chaque post-it correspondant à une tâche / *users-stories*.



## 3.2 La gestion de notre projet

L'organisation de notre projet s'est faite par une réunion regroupant le *sprint review*, la *rétrospective* et le *sprint planning*. Dans cette réunion le *product owner*, M. Gonzalez-Rubio, était présent afin de vérifier d'abord le rendu des *users-stories*. Pendant ces vérifications, il nous demandait d'ajuster certaines fonctionnalités qui ne correspondaient pas exactement à ce que le client désirait. Ensuite, il nous annonçait le *sprint backlog* définissant les tâches à réaliser pendant le *sprint*, auxquelles nous devions ajouter les modifications des *users-stories* non totalement conformes aux exigences du client. De retour dans notre bureau, chaque membre choisissait une *users-stories* qu'il allait réaliser. Puis, lorsque celle-ci était terminée, il venait en choisir une autre sur le tableau (Photo 1).

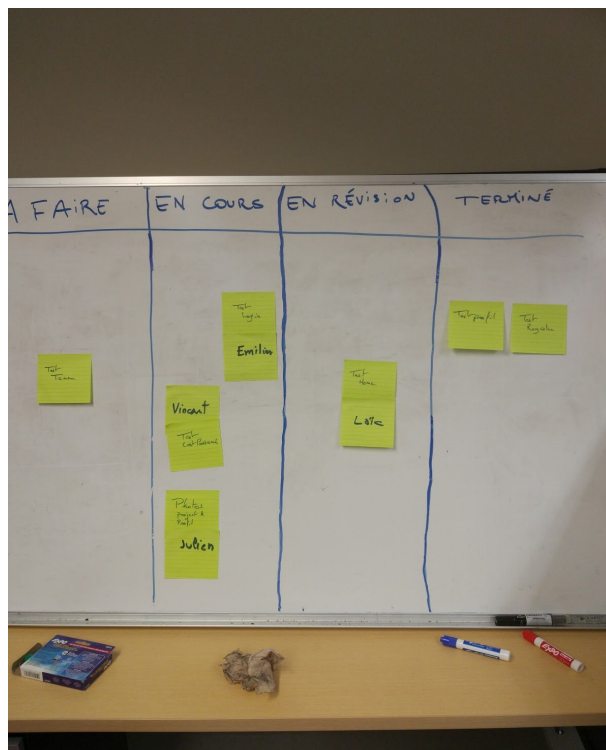


Photo 1 : tableau fin sprint numéro trois.

Les *daily scrum* n'ont pas été réalisées à proprement parler. Celles-ci étaient remplacées par les discussions que nous avons entre nous dans la journée lors de conversations informelles. En effet, l'avantage d'être un petit effectif nous permettait

de savoir sur quoi travaillait chaque personne chaque jour. De plus si quelqu'un rencontrait un problème, l'équipe entière en était alertée en peu de temps.

### 3.3 Un long apprentissage

Les outils utilisés pendant ce stage nous ont été imposé par notre tuteur de stage. Il nous a demandé d'utiliser le framework web Play permettant d'écrire rapidement des applications web en Java ou Scala. Java ne fut pas un problème pour moi puisque l'apprentissage de ce langage prenait une part importante dans ma formation. Ce n'était cependant pas le cas de Julien Beuve et Loïc Travaillé qui n'avaient pas reçu de formation sur la programmation orientée objet, causant donc une longue période d'apprentissage pour eux. Scala demanda à toute l'équipe une période d'apprentissage car inconnu pour nous quatre. Elle possède l'avantage d'avoir une syntaxe accessible inspirée du langage Ruby mais aussi la robustesse de Java.

Dès le début du stage, nous avons été alerté que notre code devra contenir des **tests**. Aucun d'entre nous n'avait idée de comment réaliser un *test* avec JUnit. Ce fut la période d'apprentissage la plus longue puisqu'elle dura la quasi-totalité du stage. La particularité d'un *test* est qu'il doit être réalisé avant le code lui-même. Cela étant frustrant pour nous qui avons l'habitude de coder tout de suite le code de production, c'est pourquoi beaucoup de *tests* furent réalisés après le code de production.

## 4. Réalisations techniques

### 4.1 Mise en place de l'environnement

L'environnement de programmation a été choisi par M.Gonzalez-Rubio. Il a porté son choix sur l'**IDE** (*integrated development environment*) Eclipse. Une chance pour moi car déjà familier avec celui-ci sur des projets hors et lors de ma formation. Cependant l'installation de divers plugins apporta quelques soucis. Cela nous retarda sur le lancement du projet. Ce problème était notamment causé par des soucis d'incompatibilité entre plusieurs de ces **plugins**.

Comme **build-tool**, **SBT** s'est logiquement imposé, étant spécialisé pour Java et Scala. Encore une fois, des problèmes sont advenus. En effet, la notion de *build-tool* était très approximative pour moi.

Qui dit projet de groupe, dit **logiciel de gestion de version**. Subversion (**SVN**) nous fut imposé à mon détriment car je suis habitué à son grand rival Git. Cependant, ce ne fut pas un grand soucis car l'ensemble du groupe utilisait correctement SVN.

### 4.2 Le modèle MVC

Le modèle MVC est un motif d'architecture logiciel orienté pour les interfaces graphiques. Comme son nom l'indique il est composé de trois modules :

- Un **modèle** gérant les données.
- Une **vue** contenant les présentations graphiques de l'interface.
- Un **contrôleur** contenant la logique des actions effectuées par l'utilisateur.

Tout notre projet est basé sur ce motif. Par exemple, lorsque l'utilisateur va appuyer sur un lien vers une page interne du site, le code va appeler une fonction

dans le *contrôleur* correspondant qui va ensuite renvoyer la *vue* adéquate. (Voir annexe 2)

## 4.3 Clean Code

Avant de démarrer le projet, notre tuteur de stage nous a demandé de lire et de résumer l'ouvrage *Clean Code* par Robert Cecil Martin. Ce livre donne et explique les conventions de code avancées pour un projet informatique. Le but de cette tâche a été de nous pousser à rendre un code propre, clair et compréhensible.

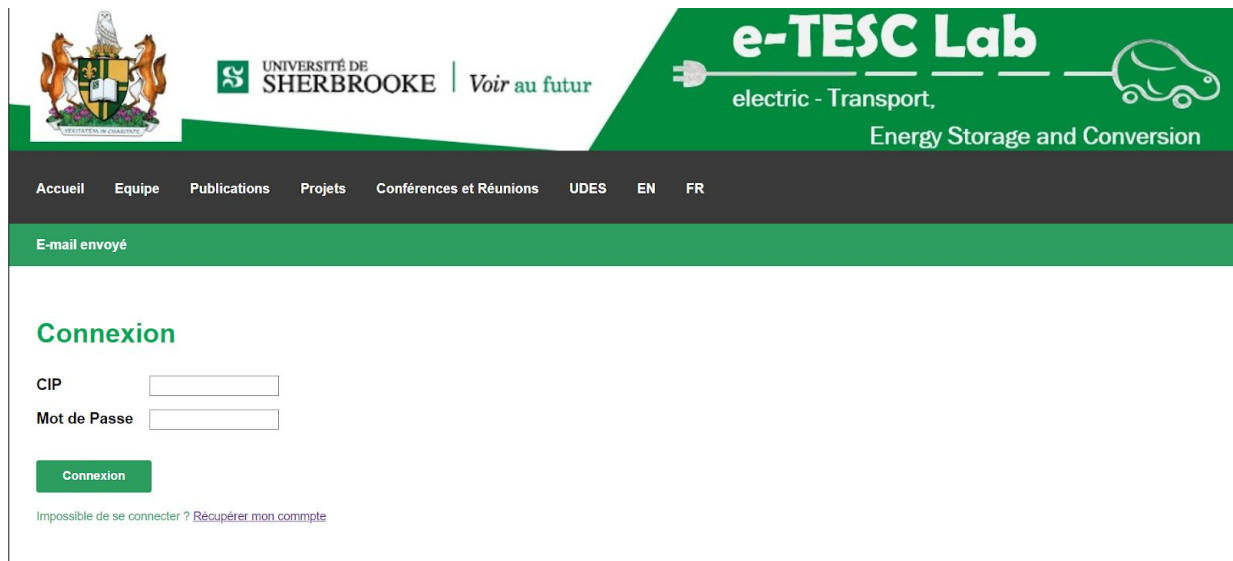
## 4.4 Envoyer un mail

Une des plus grandes et intéressantes parties du projet que j'ai réalisé seul est la conception de la fonction " mot de passe oublié " ; c'est-à-dire l'envoi d'un e-mail à un utilisateur ayant demandé son envoi, e-mail contenant un lien pour modifier son mot de passe lorsqu'il a été oublié. J'ai choisi cette manière de faire pour sa sécurité, car, en effet, de nombreux sites se contentent d'envoyer un nouveau mot de passe, ce qui est dangereux pour la sécurité des données de l'utilisateur.

Le fonctionnement de cette fonction se base sur la création d'un objet *LostPassword* qui sera créé lorsque l'utilisateur demandera l'envoi de l'e-mail. Celui-ci contient un code de vingt caractères (le token), la date de création plus cinq minutes, et l'identifiant du compte demandant l'envoi de l'e-mail.

L'utilisateur recevra donc un e-mail contenant un lien ressemblant à [www.siteweb.ca/ChangePassword/codedevingtcaractères](http://www.siteweb.ca/ChangePassword/codedevingtcaractères) . Ce lien le redirigera vers une page lui permettant de modifier son mot de passe, si la date n'est pas dépassée. En résumé, le lien est valide cinq minutes. La conception de l'envoi de le-mail en lui même ne prit pas de temps à être coder car j'ai trouvé une documentation et des tutoriels bien conçus, notamment sur l'utilisation du **package** javax.mail. Au final, l'envoi d'e-mail fonctionne immédiatement grâce à une longue période de recherche. Le seul problème rencontré est venu plusieurs semaines après l'implémentation de cette fonction. Le debuggage fut long (une demie journée) mais la conclusion fut

étonnant, en effet les e-mails ne s'envoyaient plus parce que l'adresse e-mail d'envoi était considéré par le fournisseur d'adresse e-mail comme un adresse dite *spam*. Vérifier le compte suffit à régler le problème. Ci-dessous (photo 2) on peut remarquer qu'un message *flash* apparaît pour indiquer que l'envoi de l'e-mail a bien été effectué. Il existe aussi des messages d'erreurs (photo 3)



The screenshot shows the e-TESC Lab website header with the University of Sherbrooke logo and navigation menu. Below the header, a green banner displays "E-mail envoyé". The main content area is titled "Connexion" and contains login fields for "CIP" and "Mot de Passe", a "Connexion" button, and a link for "Impossible de se connecter ? Récupérer mon compte".

Photo 2 : page après envoi de l'e-mail.



The screenshot shows the e-TESC Lab website header. Below the header, a red banner displays "E-mail introuvable". The main content area is titled "Impossible de se connecter ?" and contains a message: "Vous avez oublié le mot de passe de votre compte ? Vous pouvez le réinitialiser ici". Below this is an "E-mail" input field and an "Envoyer l'e-mail" button.

Photo 3 : page quand l'adresse e-mail indiquée n'existe pas dans la base de donnée.

La photo 4 ci-dessous montre ce qu'un utilisateur reçoit par e-mail après une requête.

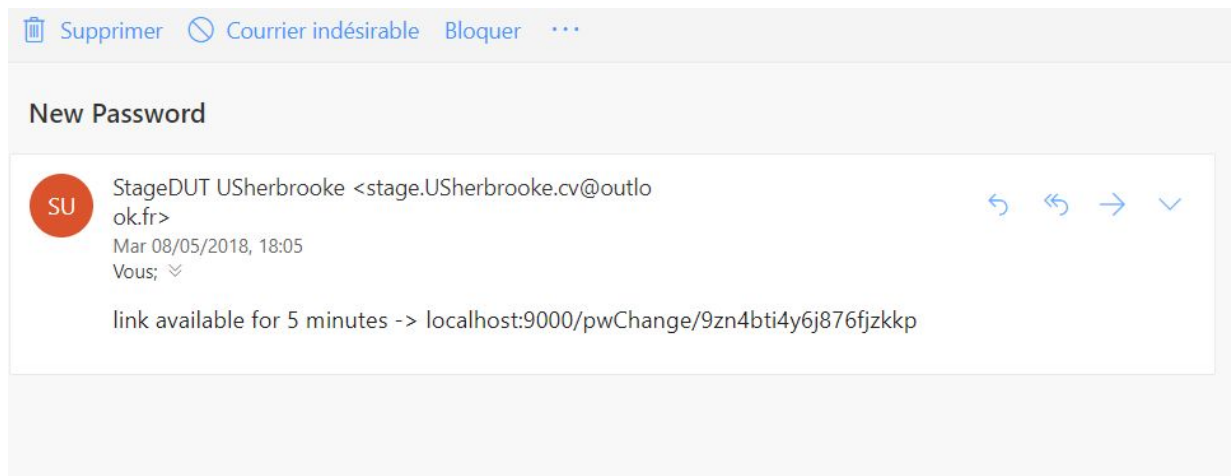


Photo 4 : e-mail de récupération de mot de passe.

## 4.5 Une page type : la page *profile*

### 4.5.1 L'accès à la page

L'accès à la page va se faire par l'appel de la méthode *profile* par le fichier route.

```
45 GET /Profile/:id controllers.ProfileController.profile(id : String)
```

Photo 5 : appel de la méthode *profile*

Cet appel est lancé par le lien présent dans la vue du menu.

```
<a href=@routes.ProfileController.profile(session.get("name")) id="mprofile"></a>
```

Photo 6 : appel dans la vue

Dans la fonction *profile*, selon le statut de l'utilisateur, l'utilisateur va être redirigé vers une page différente. La différence entre ces pages se trouve sur les différents droits, par exemple le droit ou non de modifier la page.

```
if(session("user") != null) {  
    switch(session("user")) {  
        case "admin": page = ok(views.html.templates.TemplateAdmin.render("Profile : "+user.getCip(),views.html.profiles.ProfileGestion.render(user, resume,status)));  
                      break;  
  
        case "owner": if(Administrator.getAdministratorByCIP(cip)==null) {  
                      page = ok(views.html.templates.TemplateOwner.render("Profile : "+user.getCip(),views.html.profiles.ProfileGestion.render(user, resume,status)));  
                      }else {  
                          flash(ConstantsMessagesFlash.ERROR,ConstantsMessagesFlash.message(ConstantsMessagesFlash.REGISTER_FORM_NOT_ADMIN));  
                          page = ok(views.html.templates.TemplateOwner.render("Profile : "+user.getCip(),views.html.profiles.Profile.render(user, resume,status)));  
                      }  
                      break;  
  
        case "user": if(session("name")!=null && session("name").equals(cip)) {  
                      page = ok(views.html.templates.TemplateUser.render("Profile : "+user.getCip(),views.html.profiles.ProfileGestion.render(user, resume,status)));  
                      }else {  
                          page = ok(views.html.templates.TemplateUser.render("Profile : "+user.getCip(),views.html.profiles.Profile.render(user, resume,status)));  
                      }  
                      break;  
  
        default :    page = ok(views.html.templates.Template.render("Profile : "+user.getCip(),views.html.profiles.Profile.render(user, resume,status)));  
                      break;  
    }  
} else {
```

Photo 7 : Tri de redirection

Comme on peut le voir sur la photo 7, être un *admin* ou être la personne à qui appartient le profil donnera accès à la page *ProfileGestion*, alors que si ce profil n'appartient pas à l'utilisateur et que l'utilisateur n'est pas un *admin*, il accèdera à la page *Profile*. La vue *ProfileGestion* a un bouton *edit* qui permet de modifier le profil, bouton absent dans la vue *Profile*. Ce schéma est présent dans la plupart des pages. Dans le *header* aussi, l'affichage est différent selon l'état connecté ou non et le statut de l'utilisateur.

## 4.5.2 Scala et les vues

Les vues sont un ensemble d'HTML et de code dynamique. Scala demande une arobase avant chaque partie dynamique. Sur la photo 8 la première ligne permet de charger les arguments envoyer par *views.html.profiles.Profile.render(user, resume, status)* dans le controleur (photo 7).

```
@(user : Users, resume : String, status : String )
@defining(play.core.PlayVersion.current) { version =>

<div class="profiles">
  <h1>@user.getFirstName() @user.getLastName.toUpperCase()</h1>
  <div class="hprofil">
    <div>
      <article>
        <h2 id="lcip"></h2><h3>:</h3>

        <p>@user.getCip()</p>
      </article>
      <article>
        <h2 id="lemail"></h2><h3> :</h3>
        <p><a href="mailto:@user.getMail()">@user.getMail()</a></p>
      </article>
      <article>
        <h2 id="lstatus"></h1><h3>:</h3>
        <p>@status</p>
      </article>
      <article>
        <h2 id="lresume"></h2><h3>:</h3>
        </article>
        <pre>@resume</pre>
      </div>

      
    </div>
  </div>
}
```

Photo 8 : Exemple de vue (Profile)



## 4.6 Les messages comme constante

Notre équipe s'est rendu compte que les messages affichées sur le site web ne respectaient pas les normes précédemment vues dans *Clean Code*. En effet, celles-ci n'étaient pas sous forme de constante et dans un fichier différent. J'ai donc créé deux fichiers *ConstanteMessage* et *ConstanteMessageFlash* contenant des lignes présentées sur la photo 9.

```
public static final String CONNECT_MESSAGE = "Connected";
public static final String DISCONNECT_MESSAGE = "Disconnected";
```

Photo 9 : Constante message

## 4.7 Les tests

Tout d'abord, pour réaliser les tests, le *framework* de *test* unitaire JUnit a été nécessaire. Pour le fonctionnement des tests, il est souvent nécessaire d'initier des objets (ici *users*). La fonction annotée par *@before* va s'exécuter avant le *test*. Ainsi on peut lancer une application et créer un *user*.

```
public class PasswordForgotTest extends WithApplication {

    private final String CIP = "PseudoAdminTest";
    private final String PASSWORD = "PasswordTest";
    private final String FIRSTNAME = "FisteNameTest";
    private final String LASTNAME = "LasteNameTest";
    private final String MAIL = "test@hotmail.fr";
    private final String CODE = "testtesttesttesttest";

    private Users userTest;
    private LostPassword lp;
    RequestBuilder request ;
    Application app;
    HttpSession session;

    @Before
    public void before() {
        app = fakeApplication();
        request = new Http.RequestBuilder();
        start(app);
        userTest = new Users(CIP,PASSWORD,FIRSTNAME, LASTNAME,MAIL,null);
        userTest.save();
        lp = new LostPassword(CODE, CIP);
        lp.save();
    }
}
```

Photo 10 : test (1/4)

Une fonction annotée par `@Test` va s'exécuter comme un *test*. Dans la photo 11, la fonction *testChangePassword* va exécuter la fonction *changePassword* et remplir le formulaire avec "map". Ensuite le *test* va vérifier que l'utilisateur est bien redirigé sur la bonne page et que le mot de passe a bien été modifié.

"`assertEquals(arg1, arg2)`" est une fonction retournant true si arg1 égal arg2.

```
@Test
public void testChangePassword() {
    HashMap<String, String> map = new HashMap<>();
    map.put("pass", "Testing");
    map.put("passVerif", "Testing");
    request = new Http.RequestBuilder().method("POST").uri(routes.PasswordForgotController.changePassword(CODE).url()).bodyForm(map);
    Result result = route(app,request);
    assertEquals("This is not the page", "/Login", result.redirectLocation().get());
    assertEquals(303, result.status());
    userTest = Users.find.byId(CIP);
    assertEquals("Testing", userTest.getPassword());
}
```

Photo 11 : test (2/4)

Le *test testSendMailResetPassword* va exécuter la fonction *sendMailResetPassword* et vérifier qu'un *LostPassword* a bien été créé.

```
@Test
public void testSendMailResetPassword() {
    HashMap<String, String> map = new HashMap<>();
    map.put("mail", MAIL);
    request = new Http.RequestBuilder().method("POST").uri(routes.PasswordForgotController.sendMailResetPassword().url()).bodyForm(map);
    Result result = route(app,request);
    assertEquals("This is not the page", "/Login", result.redirectLocation().get());
    assertEquals(303, result.status());
    assertNotNull(LostPassword.findByCip(CIP));
}
```

Photo 12 : test (3/4)

Une fonction annotée *@After* s'exécutera après le dernier test. Ainsi, elle est utilisée pour ne pas garder de trace dans la base de donnée des *tests*.

```
@After
public void teardown() {
    userTest.delete();
    lp.delete();
    stop(app);
}
```

Photo 13 : test (4/4)

# Conclusion

Ce stage effectué grâce au professeur M. Gonzalez-Rubio avait pour but initial de créer un site web de mise en ligne de CV. Cependant, un site web identique existait déjà (<https://ccv-cvc.ca>). C'est pourquoi notre sujet de stage a mué. L'objectif était donc de créer un site web vitrine des groupes de chercheurs de l'université.

Étant en méthode agile, les tâches à accomplir nous étaient dévoilées au fur et à mesure de l'avancement du projet. Les objectifs principaux ont été réalisés (création et gestion de profil, connexion, déconnexion, gestion d'équipe, navigation entre les projets). Un objectif donné n'a cependant pas pu être terminé à temps : celui de l'intégration de photo dans le site. La plupart des problèmes rencontrés dans ce projet sont liés au manque de connaissance sur un outil, un langage ou une méthode. Mais ces problèmes ont pu être résolus ou contournés, me fournissant des moments parmi les plus instructifs du stage.

Outre une grosse progression en langage Java, j'ai pu approcher le langage Scala, une expérience très enrichissante. Il en va de même pour la manipulation d'outils inédits pour moi comme *SBT*. Ce stage m'a permis de terminer ma formation sur une note très empirique. De plus, j'ai eu l'occasion de côtoyer beaucoup de chercheurs et doctorants me donnant la possibilité d'approcher de près leur métier. Cependant, je suis conscient que mon stage ne reflète pas l'exactitude d'une expérience en entreprise. Malgré cela, je porte un regard très positif sur ce stage qui m'a permis de progresser techniquement. Je souhaiterais dans un futur proche effectuer un stage dans une entreprise pour compléter cette expérience .

# Résumé

J'ai effectué un stage du 16 avril 2018 au 29 juin 2018. Mon tuteur M. Gonzalez-Rubio m'a accueilli à l'université de Sherbrooke pendant 11 semaines. Nous étions quatre stagiaires formant l'équipe chargée de réaliser le prototype d'un site web de mise en ligne de CV pour les chercheurs et professeurs de l'Université de Sherbrooke. Cependant, le site web existait déjà, c'est pourquoi notre sujet de stage a évolué. Notre sujet de stage final était de créer un prototype de site web vitrine dynamique pour présenter et gérer les groupes de recherche de l'université.

L'Université de Sherbrooke est une université francophone située au Canada dans la province du Québec. L'université possède huit facultés chacune dirigée par un doyen ou une doyenne, et un total de 40 000 étudiants en 2016. L'université nous a fourni, à moi et aux autres membres de l'équipe du projet, une place dans un open-space dans la Faculté de Génie.

Le projet en lui-même consistait à créer un site web dynamique ayant pour but de présenter les différents groupes de recherche présents à l'Université de Sherbrooke.

Les tâches à réaliser étaient multiples : la mise en place d'un système de connexion avec des comptes, une page de gestion et de présentation d'équipe de recherche, et la gestion des comptes.

Pour réaliser ce projet nous avons utilisé la méthode SCRUM. Chaque semaine, nous nous sommes regroupés pour effectuer une réunion contenant le *sprint review*, la *rétrospective* et le *sprint planning*. À l'issue de cette réunion, l'équipe avait connaissance du *sprint backlog*. Chacun réalisait une *users-storie* puis en réalisait une nouvelle lorsque la précédente était terminée.

Des connaissances nouvelles m'ont été nécessaires pour mener à bien ce projet ; j'ai dû apprendre à utiliser le langage Scala, à comprendre et utiliser un *build-tool* *SBT*, le framework de *test* unitaire JUnit et le framework Play pour créer des applications web.

Pour ce qui est de ma partie technique, j'ai réalisé le système d'envoi d'e-mail et réinitialisation de mot de passe, la fonction mot de passe oublié. J'ai aussi lu et résumé l'ouvrage *Clean Code* dans le but de m'imprégner des conventions de code propre. De plus, j'ai codé les pages de profils et de modification de profils ainsi que les tests pour l'envoi d'e-mail et modification de mot de passe.

# Abstract

I did an internship from April 16, 2018 to June 29, 2018. My tutor Mr. Gonzalez-Rubio welcomed me to the University of Sherbrooke for 11 weeks. We were four interns forming the team in charge of producing a prototype of an online CV website for researchers and professors at Sherbrooke University. However, the website already existed, which is why our internship topic has evolved. Our final internship topic was to create a prototype dynamic showcase website to present and manage the university's research groups.

The University of Sherbrooke, a french university, is located in Canada in the province of Quebec. The university has eight faculties each headed by a dean, and a total of 40,000 students in 2016. The university provided me with a place in an open space in the engineering faculty.

The project itself consisted in creating a dynamic website to present the different research groups present at the University of Sherbrooke. The tasks to be carried out were multiple as the setting up of a system of connection with accounts, a page of management and presentation of research team, and the management of accounts.

To realize this project we used the SCRUM method. Each week, we gathered for a meeting containing the *sprint review*, the *retrospective* and the *sprint planning*. At the end of this meeting, the team was aware of the *sprint backlog*. Everyone made a *user-story* and then made a new one when the previous one was finished.

I had to learn to use the Scala language, to understand and use a *build-tool* SBT, the JUnit unit test framework and the framework Play to create web applications.

As for my technical part, I realized the email system and password reset, the forgotten password function. I also read and summarized the book *Clean Code* with

the aim of getting acquainted with the conventions of clean code. In addition, I coded the profile and profile modification pages as well as the tests for sending emails and changing passwords.



# Glossaire

**Sprint / Itération** : période de deux à quatre semaines durant laquelle une partie fixe du projet est à réaliser.

**User-storie** : tâche à réaliser, souvent courte et indépendante.

**Sprint backlog** : liste d'*user-storie* composant le contenu d'un *sprint*.

**Product backlog** : ensemble des user-stories définie dans un projet.

**Product owner** : représentant du client, il porte la vision globale du projet.

**Scrum-master** : encadre et protège l'équipe des événements extérieurs.

**Sprint planning** : réunion où l'ensemble des parties prenantes du projet vont mettre en place le *sprint backlog*.

**Dailys scrum** : réunion hebdomadaire où chaque membre de l'équipe va rendre compte de son avancement, ses problèmes et ses missions futures.

**Feedback** : remontée d'informations client relative à sa satisfaction à l'égard d'un produit ou service.

**Sprint review** : compte rendu au client ou *product owner* du résultat du *sprint* lui permettant de recueillir le *feedback* client.

**Rétrospective** : permet aux membres de l'équipe de rendre compte des difficultés ou forces de l'équipe.

**Test** : procédure permettant de vérifier le bon fonctionnement d'une partie du code.

**Plugin** : apporte de nouvelles fonctionnalités à un logiciel.

**Build-tool** : programme qui automatise la création d'un exécutable d'application depuis le code source.

**Open-source** : programme informatique dont le code source est distribué sous une licence permettant à quiconque de lire, modifier ou redistribuer ce logiciel.

**SBT** : build-tool open-source.

**Logiciel de gestion de version** : permet de modifier, ajouter, et partager le code au sein d'un groupe.

**SVN** : logiciel de gestion de version distribué par Apache.

**Modèle** : fichier gérant les données.

**Vue** : fichier contenant les présentations graphiques de l'interface

**Contrôleur** : fichier contenant la logique des actions effectuées par l'utilisateur.

**Package** : fichier contenant des méthodes déjà définies.

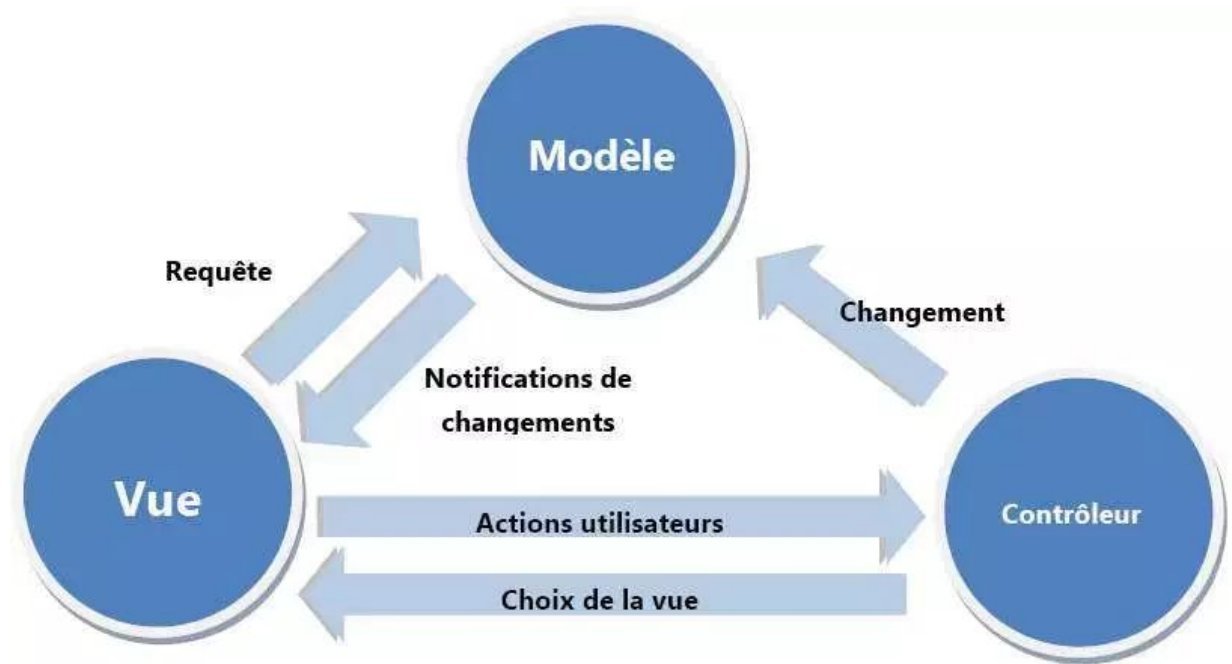
## Table des matières des annexes :

1. Tableau caractérisation d'une organisation	32
2. Schéma model MVC	33

## 1. Tableau caractérisation d'une organisation

Forme	Entreprise
Type d'organisation	Privée
Statut juridique	Personne morale sans but lucratif (OSBL)(CA)
Finalité	Réaliser des profits pour assurer sa pérennité
Objectifs	<ul style="list-style-type: none"> <li>- Reconnaissance internationale</li> <li>- Extension du nombre de formation</li> </ul>
Nationalité	Canadienne
Activité principale	Education
Secteur d'activité	Tertiaire
Taille	6 778 employés
Origine des capitaux	na
Besoin	Non-solvable
Domaine d'intervention ou champ d'action	Education : National Recherche : International
Ressources	571,2 millions \$CA
Mode de coordination	Supervision directe
Mode de direction	Assemblée générale

## 2. Schéma model MVC



A partir du site <http://thegalsengeek.com>