
Développement d'un site Web



Laboratoire eXit – Université Sherbrooke

Étudiant : Julien Beuve
Responsable de Stage : Ruben Gonzalez-Rubio
Tuteur IUT : Hervé Gauvrit

Remerciements

Je tiens à remercier toutes les personnes qui m'ont permis de réaliser ce stage dans les meilleures conditions possible :

Ruben Gonzalez-Rubio, mon maître de stage et directeur du laboratoire eXit. Il nous a aidé tout au long de notre séjour au Québec notamment pour le logement. Je tiens également à le remercier pour m'avoir donné cette opportunité et pour ses conseils.

Claire Constantin, professeur d'anglais à l'IUT de Rennes, pour m'avoir proposé ce stage au Québec, et pour l'aide qu'elle m'a apportée dans les démarches administratives.

Julie Giguère, chargée de l'affaire académique qui nous a transmis toutes les démarches administratives à faire lors de notre arrivée au Québec.

Loïc Tracailly, Vincent Le Page et Emilier Cosnier, les trois étudiants en stage au laboratoire eXit, pour leur présence et leur aide aussi bien dans le cadre du stage que dans les activités faites ensemble au cours de ces quelques semaines.

Introduction :

Du 9 avril au 22 juin 2018, j'ai effectué mon stage au sein du laboratoire eXit qui est rattaché à l'Université de Sherbrooke au Canada. M. Gonzales-Rubio, est le fondateur de ce laboratoire spécialisé dans le domaine de l'informatique. Le laboratoire eXit est orienté dans la recherche et le développement de logiciels. Désireux d'acquérir de nouvelles connaissances dans le domaine de l'informatique, j'ai choisi un stage en programmation et développement. Le laboratoire eXit. La possibilité d'effectuer ce stage à l'étranger permet également de découvrir d'autres méthodes de travail.

Notre stage a pour but de réaliser une première ébauche afin que le projet soit présenter devant un conseil qui décidera par la suite si ils adhèrent au projet ou pas. Une fois approuvé, une équipe de développement est mise en place pour mener à bien ce projet.

Notre premier sujet consistait à réalisé un site internet où les professeurs de l'université pouvait ajouter ou modifier leurs CV. Tous les autres utilisateurs avaient la possibilité de consulter les CV autres que les leurs. Pour réaliser cette mission nous avons travaillé en groupe de quatre personnes Loïc Travaillé, deux étudiants en DUT Informatique à Lannion et moi-même. Pour mener à bien cette première mission nous avons dû nous documenter sur les différents outils que nous allions utilisés.

Par la suite le projet à changer, le sujet portait toujours sur la création d'un site Web pour l'Université de Sherbrooke mais l'utilisation de cette dernière à changé. Tout d'abord il faut savoir que l'université possède plusieurs laboratoire, chacun d'eux regroupe plusieurs projets et plusieurs membres . Notre mission était de réalisé un site de gestion de projet, permettant à toute personne de se renseigner sur les projets en cours, les membres et leurs rôles au sein de l'université (étudiant, professeurs, doctorant, etc). Les membres inscrit du site pouvaient éditer ou ajouter des projets selon leurs statuts sur le site.

Dans un premier temps, je vais vous présenter le cadre de notre stage, l'Université de Sherbrooke. Par la suite je parlerais des différents outils que nous avons utilisés pour mené à bien les différentes missions qui nous ont été confiées. Puis j'aborderai les différentes tâches que nous avons développé, sur chaque page je développerai dans l'ordre d'évolution des sprints. Enfin je terminerai par ma conclusion sur mon stage aussi bien technique que personnelle,

Table des matières

Présentation.....	6
A Présentation de l'Université.....	6
B Laboratoire eXit.....	7
Présentation du Projet.....	8
A Notre Projet.....	8
1 Cahier des Charges.....	8
Outils utilisés.....	9
A Java.....	9
1 Langage Orienté Objet.....	9
2 Les Classes.....	9
3 Les packages.....	9
B Scala.....	10
C JavaScript.....	10
D HTML.....	10
E CSS.....	11
F Eclipse.....	11
G SVN.....	12
H TortoiseSVN.....	13
I Méthode Scrum.....	13
1 Répartition des rôles dans Scrum.....	13
2 Le Product Backlog.....	14
3 Le Sprint Backlog.....	14
4 Les Sprints.....	15
J Architecture MVC.....	15
Réalisation du Projet.....	17
A Démarrage.....	17
1 Premier sprint.....	18
2 Deuxième sprint.....	19
3 Troisième sprint.....	20
B Clean Code.....	20
C Template.....	20
D Page Home.....	21
E Page Login.....	23
F Page Récupération.....	25
G Page Projet.....	25
H Page MyProfil.....	26
I Page Gestion Profil.....	27
J Page Création de Compte.....	27
K Page Team Active.....	28
L Add Project.....	29
M Add Team list.....	30
N Traduction Anglais/Français.....	31
O Autre Page.....	32
P Les tests.....	33
Conclusion.....	35

A Conclusion technique.....	35
B Conclusion Personnelle.....	36

Présentation

A Présentation de l'Université

L'Université de Sherbrooke est une université francophone qui a été fondée en 1954 et est située dans la région de l'Estrie qui aujourd'hui compte plus de 40 000 étudiants. Elle est répartie sur quatre campus : le campus de Longueuil, le campus de la santé et le campus de la recherche. Notre stage a eu lieu sur le campus principal de l'Université de Sherbrooke.



Les enseignements proposés et ses ressources mises à disposition ont fait d'elle l'université francophone la plus appréciée par ses étudiantes et étudiants au Canada. L'université est également réputée pour son régime coopératif de stages qui lui permet de faire des échanges universitaires à l'international. Grâce à cela, on compte plus de 90 pays différents parmi ses 40 000 étudiants.

B Laboratoire eXit



Le laboratoire eXit a été créé par Ruben Ganzalez-Rubio, il fait partie de la Faculté des Génies appartenant à l'Université de Sherbrooke. Les membres qui le composent sont principalement des doctorants ainsi que des stagiaires . La fonction de ce laboratoire est d'actualiser ou encore faciliter l'utilisation de certain logiciel tout en respectant un cahier des charges avec des contraintes techniques strictes.

Présentation du Projet

A Notre Projet

1 Cahier des Charges

Notre stage avait pour but de réaliser une ébauche de site internet pour l'Université de Sherbrooke. L'utilisation de ce site a évolué au cours de notre stage, c'est pourquoi nous avons eu deux cahiers des charges différents.

Pour le premier, nous devions programmer à l'aide de différents langages un site Web offrant la possibilité à tous les enseignants d'ajouter, de modifier et consulter leurs CV.

Toujours sur la création d'un site Web, le deuxième cahier des charges avait pour but de réaliser un site de gestion de projet permettant à tout utilisateurs de se renseigner sur les différents projet en cours et leurs membres. Le site devait offrir d'autres fonctionnalités aux utilisateurs possédants un compte. Ce site existait déjà pour le laboratoire e-TESC mais il était programmé de façon statique, notre mission était de développer une version dynamique de ce site.

Outils utilisés

A Java

Java est un langage de programmation orienté objet créé par des employés de Sun Microsystems en 1982. Le but de ce langage est d'être facilement portable sur plusieurs systèmes d'exploitations tel que Windows, Linux ou encore Mac ; il est également utilisé dans beaucoup de domaines comme la robotique mobile ou l'informatique.

1 Langage Orienté Objet

Il s'agit d'un modèle de programmation informatique inventé par les Norvégiens dans les années 1960. Son principe est simple, le programme est partagé en plusieurs bloc, chaque bloc correspond à une idée ou une entité physique comme par exemple une maison. Il possède une structure interne et des liaisons avec ses pairs, il s'agit de représenter ces objets et leurs relations. C'est grâce aux liaisons établies entre les objets que nous pouvons réaliser des fonctionnalités qui nous permettront de résoudre des problèmes.

2 Les Classes

Une classe regroupe des méthodes et des attributs. Les méthodes servent à représenter le comportement de l'entité physique, quand aux attributs ils représentent son état.

3 Les packages

Pour organiser notre projet de façon efficace, nous utilisons des packages où les classes sont répertoriées. Certaines méthodes définissent la manière d'organiser ces packages selon le rôle des classes. Par exemple les classes qui ont pour rôle le stockage de données seront placées dans le même package.

B Scala

Scala est un langage de programmation multi-paradigme. Conçu à l'Ecole polytechnique fédérale de Lausanne en 2003, ce langage permet aux développeurs d'exprimer les modèles de programmations dans une forme concise et élégante. Son nom vient de « scalable language » ce qui veut dire « langage adaptable », en effet Scala intègre les paradigmes de programmation orientée objet et de programmation fonctionnelle. Le fait de concilier ces deux paradigmes habituellement opposés permet aux développeurs de choisir lequel des deux est le plus approprié pour résoudre son problème. Ce qui a fait la force de ce langage, c'est également la diversité de ses bibliothèques car il a la possibilité d'utiliser des bibliothèques écrites en Java.

C JavaScript

Le JavaScript est un langage de programmation orienté objet développé par Netscape en 1995. Sa principale utilisation est à l'intérieur des pages Web, aujourd'hui presque toutes les pages Web contiennent du JavaScript. Il apporte aux pages Web des fonctionnalités correspondant à des besoins spécifiques, par exemple il permet de faire défiler un texte, insérer un menu dynamique, ouvrir des fenêtres pop-up, etc. Au cours de notre projet nous l'avons utilisé pour différents besoins liés à l'affichage de nos pages Web.

D HTML

Le langage HTML est très connu, il s'agit d'un langage de description de document utilisé sur internet pour la création des pages Web. Le sigle HTML signifie « Hyper Text Markup Language », le balisage HTML est incorporé dans le texte du document et est interprété par le navigateur Web. Ce langage est simple à comprendre grâce à son système de balise qui ont chacune un rôle, une fonctionnalité comme par exemple afficher un titre ou encore un bouton.

E CSS

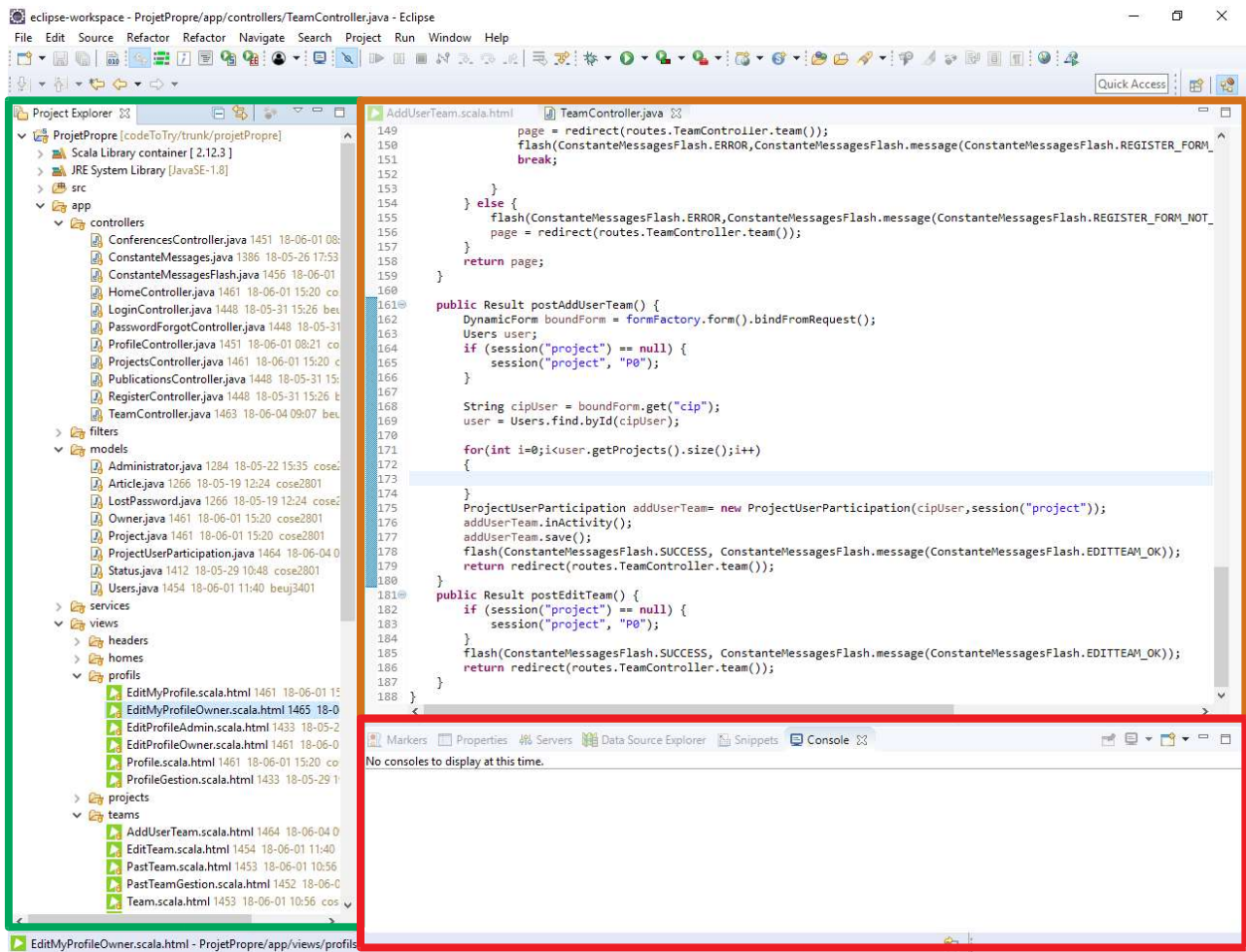
Le langage Cascading Style Sheets plus communément appelé CSS, est lié au langage HTML. En effet ce langage à pour but de modifier l'apparence de la page créer à l'aide du langage HTML. Grâce à lui, le langage HTML ne sert qu'à définir la structure des contenus. Les CSS permet de créer des styles et un design général du site.


F Eclipse

Lorsque l'on programme, il y a plusieurs étapes. Dans un premier temps il faut écrire le programme dans un éditeur de texte, une fois le programme écrit il va falloir le compiler pour qu'il se lance, on appelle cela un compilateur. Une fois lancé, des bugs peuvent apparaître, pour corriger l'erreur il existe des débogueurs qui vont nous aider à la trouver. Vous vous doutez bien qu'il sera très embêtant de devoir utiliser trois logiciels différents à chaque fois que l'on veut programmer.

C'est pourquoi il existe des IDE (Integrated Development Environment) qui possède tout le nécessaire pour programmer, il y aura un éditeur de texte, un compilateur et un débogueur intégré dans le logiciel. De plus, lors de notre projet nous avons travaillé avec plusieurs langage tels que Java, Scala, HTML, CSS, JavaScript ; Eclipse nous a permis de programmer dans tout ces langages dans un seul éditeur de texte.

Voici comment se présente l'interface d'Eclipse :



 Dossiers de nos projets

 Éditeur de texte

 Console de débogage

G SVN

Subclipse SVN est un plugin téléchargeable permettant de travailler en groupe sur un même projet. Pour cela le projet est localisé sur un serveur où tous les membres de l'équipes peuvent venir le modifier sans devoir passer par de nouvelle version de projet qui devienne trop nombreuse ce qui nuit à l'avancement du projet.

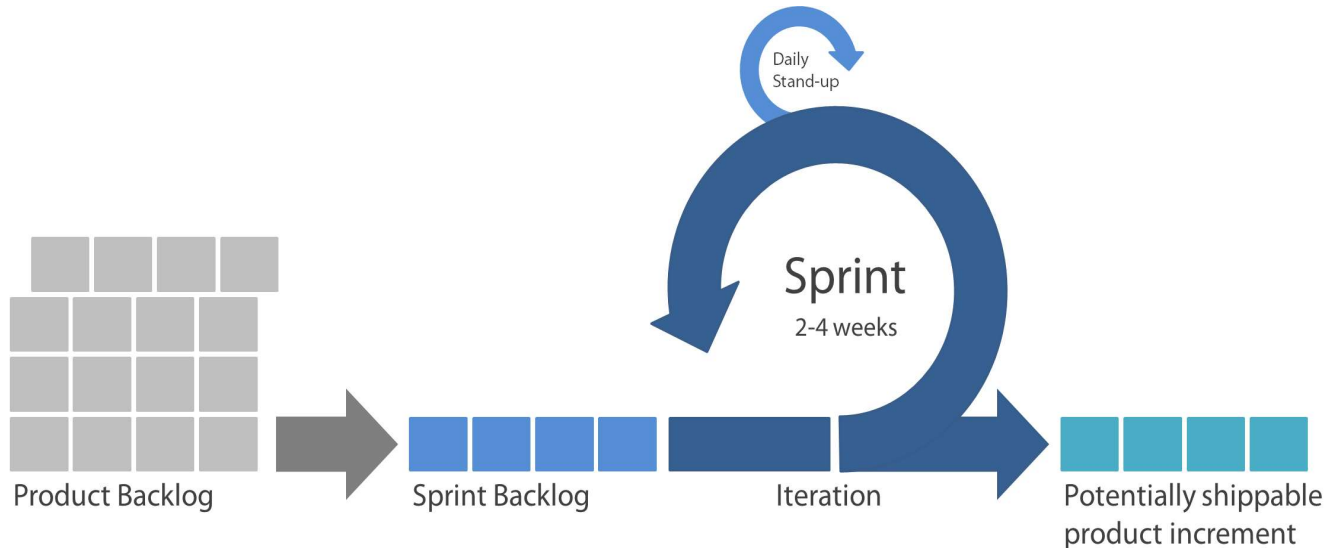
H TortoiseSVN

Il s'agit d'un logiciel libre distribuer sous forme de plugin. Il permet à un utilisateur d'avoir une interface graphique permettant de réaliser la plupart des tâches qu'offre SVN en ligne de commande.

I Méthode Scrum

La méthode Scrum apporte un cadre de travail permettant une gestion de projet dites «agile ». Elle à pour but d'améliorer la productivité de l'équipe.

Un projet agile est organisé en cycles de développements itératifs et incrémentaux. L'objectif n'est plus de mener le projet à bien, mais de donner naissance à un produit fini, qui correspond parfaitement aux besoins du client. Cette méthode ne dit comment réussir son projet ni comment surmonter les obstacles, il se contente d'offrir un cadre de gestions de projet : les rôles, un rythmes de travail, des règles et des réunions précises et limitées dans le temps.



Afin de comprendre le schéma ci-dessus, il important d'expliquer les différentes étapes :

1 Répartition des rôles dans Scrum

- **Le Scrum Master**
 - S'assure que le principe et les valeurs de Scrum sont respectées

- Facilite la communication au sein de l'équipe
- Cherche à améliorer la productivité et le savoir-faire des on équipe

- **L'équipe**

- Développeur et testeurs du logiciels
- Tous les membres de l'équipe apporte leurs connaissances pour accomplir les tâches
- Une équipe est composée de 6 à 10 personnes en moyenne et pouvant aller jusqu'à 200 personnes

- **Le Product Owner**

- Expert métier, définit les spécifications fonctionnelles
- Établit la priorité des fonctionnalités à développer ou corriger
- Valide les fonctionnalités développées
- Joue le rôle du client
- Créer le « Product Backlog » qui contient toute les fonctionnalités demandé dans le logiciel.

2 Le Product Backlog

Le Product Backlog pourrait être définit comme « l'ensemble des fonctionnalités du produit que l'on veut développer », il s'agit des attentes du Product Owner. Il priorise les fonctionnalités, afin d'implémenter en premier celle qui a le plus de valeur.

3 Le Sprint Backlog

Le Sprint Backlog représente toutes les tâches qui ont été sélectionnées depuis le Product Backlog afin d'améliorer le produit ou d'ajouter de nouvelles fonctionnalités. Il fait le liens entre le Product Backlog et le Sprint. Lors de cette étape, l'équipe doit décider ensemble de la difficulté de chaque tâches. La difficulté est noté entre 1 et 10 avec 1 très facile et 10 très compliqué, chaque membre donne un chiffre correspondant à la facilité de réalisation, si certains membres ne sont pas d'accord sur le chiffre ils doivent débattre jusqu'à ce que l'un deux rectifie. Ensuite une fois que l'équipe à juger toutes les tâches, elle décide du temps nécessaire à la réalisation, évidemment elle varie selon le niveau de difficulté de la tâche.

4 Les Sprints

Le cycle de vie Scrum est rythmé par des courts projets de quelques semaines, les sprints. Chaque sprint est constitué de plusieurs tâches à effectuer par les développeurs. Chacun d'eux ont une tâche qu'ils doivent mener à bien avant de passer à la suivante. Au cours des sprints, de courtes réunions d'avancement sont organisées entre tous les membres de l'équipe. Le but est de s'assurer que les objectifs seront tenus, cette étape est appelée le « Scrum ». Cette réunion permet également de stimuler l'esprit d'équipe et l'engagement de chaque membre de l'équipe.

« photo tableau de la méthode scrum »

Un Sprint contient 4 colonnes distinctes :

- To Do : ce sont toutes les tâches restantes
- Doing : ce sont toutes les tâches en cours de développement
- To Review : ce sont les tâches qui sont terminées mais qui doivent être vérifiées par un autre développeur (respect des contraintes, bon fonctionnement, etc)
- Done : Les tâches qui sont terminées et vérifiées

Chaque colonne contient des tâches déterminées par le Sprint Backlog.

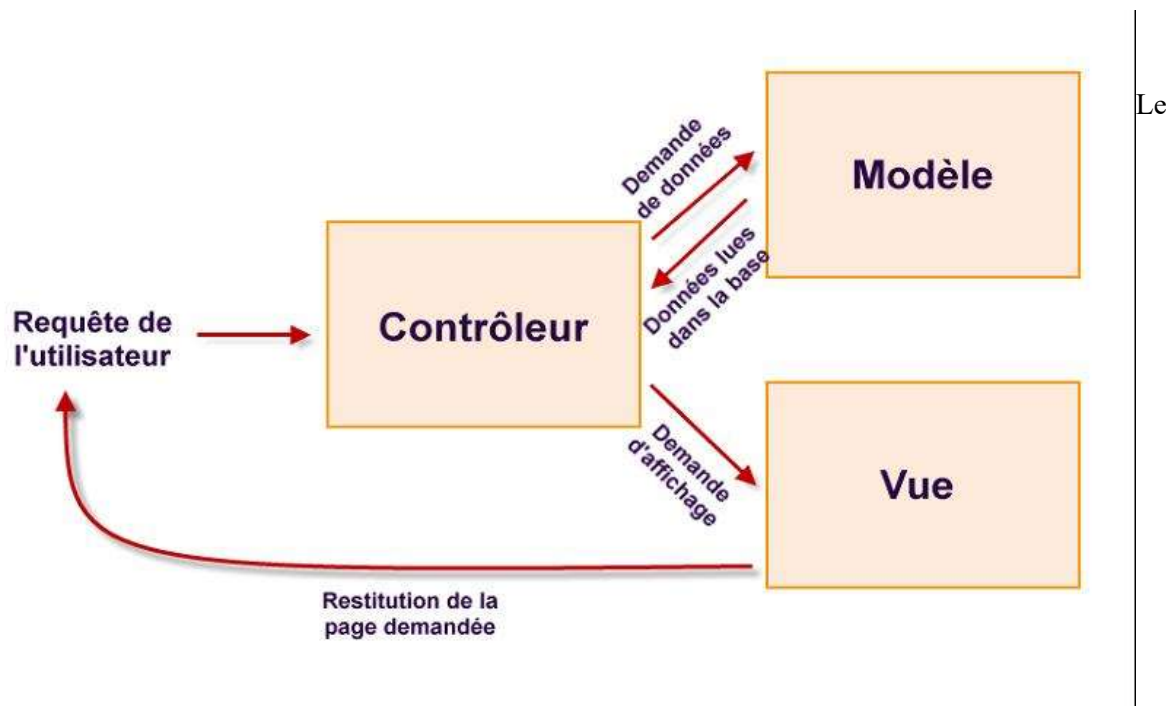
La méthode de travail Scrum permet de savoir l'avancement du projet et ainsi d'avoir une certaine efficacité dans la création de projet.

J Architecture MVC

L'architecture MVC fait partie des plus célèbres designs patterns. MVC signifie **Modèle - Vue – Contrôleur**. Le pattern MVC permet d'organiser correctement son code source. Il permet aux développeurs de se faire une idée des fichiers qu'il faudra créer par la suite, mais surtout de définir leurs rôles au sein du programme. Son but est de séparer la logique du code en trois parties distinctes :

- **Modèle** : cette partie sert à gérer les données du site que l'on veut créer. Son rôle est de récupérer les informations dans la base de données, de les organiser et de les assembler pour qu'elles puissent ensuite être traitées par le contrôleur. On y trouve entre autres les requêtes SQL.
- **Vue** : cette partie se concentre de l'affichage. Dans la vue il n'y a presque aucune condition ni de calcul, elle se contente de récupérer des variables pour savoir ce qu'elle doit afficher. On y trouve principalement du code HTML mais on peut également y trouver des boucles et conditions PHP très simples, par exemple pour afficher une liste de personnes.
- **Contrôleur** : cette partie gère la logique du code qui se charge de prendre des décisions. C'est en quelque sorte l'intermédiaire entre le modèle et la vue : le contrôleur va demander au modèle les données, il va ensuite les analyser puis prendre des décisions pour enfin renvoyer le texte à afficher à la vue. Dans notre projet le contrôleur est exclusivement du JAVA.

La figure suivante montre sous forme de schéma le mode de fonctionnement de l'architecture MVC :



Le contrôleur est celui qui dirige le programme, c'est lui qui reçoit la requête du visiteur et qui contacte la vue et le modèles pour échanger des informations avec eux afin de mener à bien la requête. Il s'occupe également des vérifications d'autorisations et des différents calculs à effectuer.

Réalisation du Projet

A Démarrage

Pour démarrer dans les meilleures conditions possibles, lors des trois premières semaines de notre stage, nous nous sommes documentés sur les méthodes à adopter lorsque l'on code. Nous avons donc lu le livre *Clean Code* écrit par Robert Cecil Martin, ce livre écrit en anglais explique les différentes étapes à respecter pour réaliser un programme propre, facile de compréhension.

Par la suite nous devons faire l'installation des différents logiciels nécessaires au projet sur nos ordinateurs. Les premières difficultés sont apparues lors de l'installation de Eclipse Jee Oxygen, les versions des différents plugins utilisés n'étaient pas toutes compatibles. Nous utilisons donc Eclipse Jee Oxygen 3A version 4.7.3a avec les plugins suivants :

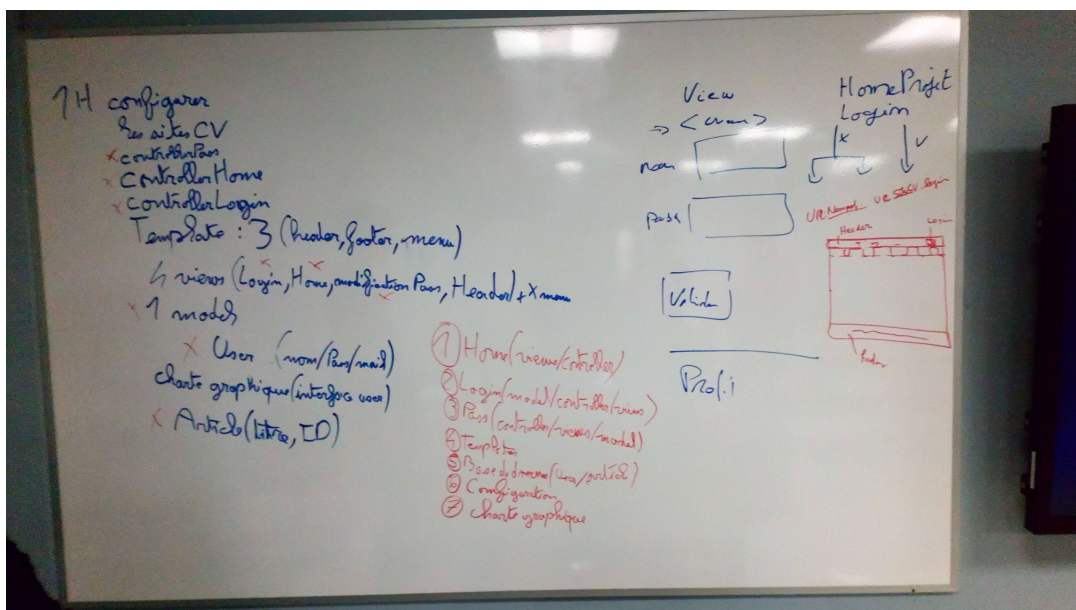
- Subclipse 4.2.3, pour importer les projets via SVN
- EcEmma Java Code Coverage 3.0.1 pour effectuer différents tests
- Scala IDE 47 pour développer en Scala

De plus, nous avons utilisé la version 1.8 de Java, version stable la plus récente, ainsi que sbt 1.1.4 notamment pour lancer le site.

Comme présenté précédemment, nous avons utilisé la méthode Scrum pour rythmer le développement. Nous avons réalisé trois sprints et commencé un quatrième mais il ne sera pas abordé dans ce rapport car il vient d'être commencé.

1 Premier sprint

Une fois tous les préparatifs terminés, nous avons établis le cahier des charges. Dans un premier temps notre projet consistait à réaliser un site internet dans lequel les professeurs de l'université pourraient trouver leurs CV. Notre tuteur nous a demandé de réaliser une ébauche contenant une page principal, une barre de menu permettant de changer de page dans le site, une page pour se connecter et un système de récupération de mot de passe si ce dernier est oublié par son utilisateur. Nous avons donc identifié les différentes Classes, Controllers, Models et Vues dont nous aurions besoin pour réaliser cette première partie. tout ceci composais le premier sprint.



Une fois cette tâche terminée, nous nous sommes fixés une date butoir à laquelle tout devait être opérationnelle, nous avons donc décidé que deux semaines seraient suffisantes pour mener à bien cette mission. Afin de travailler efficacement et de manière organisée, nous avons travaillé sur le modèle de la méthode Scrum vue précédemment. Dans un premier temps nous avons travaillé sur les tâches principales pour avoir l'architecture du site. Je me suis principalement occupé de la page Home, la conception de cette page n'était pas très compliquée, car au début du projet je devais juste afficher une page vide. La plupart des sites Web sont constitués de plusieurs pages, chacune d'elles a un but bien précis. Par exemple la page Home sert de page d'accueil lorsqu'un visiteur arrive sur le site, la page Edit my CV sert tout simplement à modifier les informations du CV. Pour faciliter la navigation entre toutes ces pages nous avons donc créé un « Template » qui possède plusieurs onglets redirigeant vers les pages correspondantes.

2 Deuxième sprint

A la fin de notre premier Sprint, notre tuteur nous a parlé d'une évolution différente du projet. L'université possède un site Web où l'on peut retrouver le nom d'un projet, le nom des personnes qui travaillent sur ce projet, et enfin un résumé du projet en question. Seulement ce site référence actuellement un seul laboratoire, le laboratoire e-TESC, il est programmé de façons statique ce qui rend difficile la mise à jour pour preuve ce site n'a pas été mis à jour depuis 2008. Notre mission est de mettre à jour les sites ci-dessous.

Groupe eXit

<ul style="list-style-type: none"> Accueil Membres Calendrier Archives Coordonnées Projets <p>Liste Publications</p>	<p>Notre vision :</p> <p>Nous considérons les systèmes d'information comme étant des systèmes offrant des services à une organisation.</p> <p>Le développement de système d'information se fait par l'ajout de services ou par la modification de services existantes aux systèmes d'information déjà en place.</p>	<p>Les principaux défis du développement d'un système d'information :</p> <ul style="list-style-type: none"> Pouvoir s'intégrer avec des systèmes déjà en place. Être à la fine pointe de la technologie du génie logiciel. Suivre les normes en vigueur pour les interfaces utilisateurs Développement rapide de nouveaux services Minimiser la perturbation des systèmes en production Système durable
--	--	---

Accueil | Membres | Calendrier | Archives | Coordonnées | Projets | [Liste Publications](#)

Espace réservé aux indications sur les droits d'auteur ou de propriété.
Pour toute question ou problème concernant ce site Web, envoyez un courriel électronique à [Ruben Gonzalez-Rubio](#).
Dernière mise à jour le : 20 October 2008.



The screenshot shows the e-TESC Lab website header with the University of Sherbrooke logo and the tagline 'Vivre au futur'. The main navigation bar includes links for Home, Team, Publications, Projects, Conferences & meetings, Press Links, and FR | EN. The main content area features a section titled 'e-TESC Lab # Under Construction #' with a paragraph about sustainable mobility. Below this, there is a section titled 'Canada Research Chair in Efficient Electric Vehicles with Hybridized Energy Storage Systems' which includes four images: a white sports car, a man in a suit, a close-up of a car's internal components, and a diagram of a hybrid energy storage system with multiple energy sources (Source 1, Source 2, Source n) connected to a central unit.

Comme lors du premier projet nous avons utilisés la méthode Scrum afin d'identifier les tâches à effectuer. Pour ce projet nous avons récupéré certaines parties déjà présentes dans notre premier Sprint.

3 Troisième sprint

Lors de ce sprint nous avons ajouter différentes fonctionnalités tels que l'ajout de projet ou encore l'ajout de participant au projet par un propriétaire. Nous avons également beaucoup travaillé sur les tests pour les rendre fonctionnels.

B Clean Code

Avant de démarrer la conception du site Web, notre tuteur nous a demandé de lire le livre *Clean Code* écrit par Robert Cecil Martin parût le 17 juillet 2008. Ce livre de 462 page en anglais explique les différentes choses à faire et à ne pas faire pour réaliser un code propre qui puisse être lu par n'importe quel autres programmeurs. Vous pouvez retrouvé ce résumé dans la partie annexe en fin de rapport.

C Template

Sur un site Web on y retrouve généralement un template. Il est composé d'un header et d'un footer. Sur notre site Web le header correspondait à un menu permettant à un utilisateur de naviguer sur le site à l'aide d'onglet cliquable. Le footer, lui, faisait office de fin de page on y retrouvait toutes sorte d'information comme par exemple le contact, l'adresse, etc.

Lors du premier sprint nous avons créé un header comportant plusieurs menu renvoyant chacun vers une page spécifique comme l'accueil, la page pour se connecter ou autre. Le footer quand à lui était juste composé des noms de l'équipe de développement c'est à dire les notres. Ces deux parties du site sont programmées en langage HTML puis mise en forme par le langage CSS. Le header est créé à l'aide d'une balise <header> qui contient une partie avec des liens renvoyant vers les différentes pages Web.

```
<header>
  
  <div class="link">
    <div class="menu">
      <a href=@routes.HomeController.home() id="mhome"></a>
      <a href=@routes.TeamController.team() id="mteam"></a>
      <a href=@routes.PublicationsController.publications() id="mpublications"></a>
      <a href=@routes.ProjectsController.projects() id="mprojects"></a>
      <a href=@routes.ConferencesController.conferences() id="mconferences"></a>
      <a href="https://www.usherbrooke.ca">UDES</a>
      <a href=@routes.HomeController.changeLanguage("english") class="language" id="en">EN</a>
      <a href=@routes.HomeController.changeLanguage("french") class="language" id="fr">FR</a>
    </div>
  </div>
</header>
```

Dans le deuxième sprint nous avons décidés de plusieurs statuts sur le site, notre tuteur nous avais demandé de trouver un moyen visuel permettant de savoir si on était connecté en tant que User, Owner ou Admin. Nous avons donc décidés d'accorder une couleur de template à chacun d'entre eux, le rouge pour l'administrateur, le bleu pour le propriétaire et enfin pas de changement de couleur pour un utilisateur normal ou pour une personne non-connecté. Pour changer la couleur de ces templates, nous testions la valeur d'une variable de session qui nous renvoyait le statut de la personne connecté, ce test a été écrit dans la vue Template.scala.

D Page Home

Notre site devait contenir une page Home qui accueillait les utilisateurs. Sur cette page, les utilisateurs pouvaient consulter les CV mis en ligne sous forme d'une liste. La liste de tout ces documents se trouvait dans notre base de donnée qui sauvegardait toutes les données des CV enregistrés sur le site.

Comme vue précédemment dans les outils utilisés, la page Home devait respecter l'architecture MVC. Dans un premier temps on voulais juste afficher une page lambda, n'ayant pas d'informations dans la base de données qui soit nécessaire à l'affichage, la page n'avais pas besoin de models. J'avais donc écrit du contenu de manière statique.

Le HomeController a été très rapide à réaliser car la page Home ne possédant pas de fonction particulière il suffisait de faire appel à la vue correspondante.

```

8 public class HomeController extends Controller {
9
10     public Result home() {
11         return ok(views.html.Home.render("Home"));
12     }
13 }
14
15 }

```

On peut voir que que j'ai seulement retourné la vue ainsi que le nom de la page qui apparaissait sur l'onglet. Le nom de la page est souligné ci-dessus en rouge.

Le texte qui se trouve sur la page avait d'abord été en statique :

```

1 @*Content Home Page*@
2
3 @defining(play.core.PlayVersion.current) { version =>
4
5     <h1> Home Page</h1>
6
7     <p>Hello my name is Julien Beuve.</p>
8 }

```

Ensuite au deuxième sprint, cette page devait varier selon le laboratoire que vous aviez choisis. Si vous choisissiez le laboratoire eXit vous seriez tomber sur le résumé d'un des projets de ce laboratoire, et inversement pour l'autre laboratoire. Une fois connecter, si votre statut était Propriétaire et que vous apparteniez au projet visible sur la page Home, un bouton «EDIT» apparaissait qui vous renvoyait sur une page où vous pouviez changer le résumé de votre projet qui était par la suite sauvegardé dans la base de donnée. Lorsqu'un projet venait d'être créer, ses résumés en anglais et en français étaient encore vide, pour les modifier il suffisait d'écrire dans un textarea qui va récupérer les données écrites pour ensuite les enregistrées dans la base de donnée. Pour ça nous avons eu besoin des librairies BoundForm et FormFactory, elles servaient à récupérer les données d'un input , d'un textarea, etc. On initialise une variable project en récupérant le projet actif grâce à une variable de session, ensuite on initialisait une variable « resume » qui avait pour but de récupérer le résumé écrit dans le textarea.

Ensuite on envoyait la variable « resume » dans le projet actif. Enfin à l'aide de la commande « .save() » le résumé était sauvegardé dans la base de donnée. Une fois que l'utilisateur avait appuyé sur le bouton « VALIDER », un message flash apparassait pour lui signaler que le projet avait bien été modifié.

```

30 public Result postedithome() {
31
32     DynamicForm boundForm = formFactory.form().bindFromRequest();
33
34     Project projettest = Project.find.byId(session("project"));
35     String resume;
36
37     if (boundForm.hasErrors()) {
38
39         flash(ConstantsMessagesFlash.ERROR, ConstantsMessagesFlash.ERROR_FORM_MESSAGE);
40         return redirect(routes.HomeController.edithome());
41
42     } else {
43
44         resume = boundForm.get("resume");
45         projettest.setResume(resume);
46         projettest.save();
47         flash(ConstantsMessagesFlash.SUCCESS, ConstantsMessagesFlash.EDITHOME_OK);
48         return redirect(routes.HomeController.home());
49
50     }
51 }
52

```

E Page Login

Dans le premier sprint cette page permettait aux utilisateurs de se connecter afin d'avoir accès à la fonctionnalité Edit pour éditer son CV. Le login était réalisé à l'aide d'un pseudo et d'un mot de passe qui étaient tout deux contenus dans la base de donnée. Si le mot de passe ou bien le pseudo était incorrect un message apparaissait afin de signaler l'erreur. Pour les personnes qui avaient oubliés leurs mot de passe, un lien de récupération est disponible. Ce lien vous redirigeait vers une autre page où vous deviez renseigner votre e-mail afin de modifier votre ancien mot de passe.

Contrairement à la page précédente, pour le login nous avons eu besoin d'un model. Le model User contenait trois fonctions :

- **la fonction formulaire** : elle servait à rediriger l'utilisateur soit vers la page Home si il s'était connecté, et reste sur la page de login si il avait échoué lors de son identification.

```
public Result loginForm() {
    if(session("name") != null) {

        return redirect(routes.HomeController.home());

    }else {

        return ok(views.html.templates.Template.render("Login",views.html.Login.render()));

    }
}
```

- **la fonction valider** : cette fonction permettait d'envoyer les informations renseignées dans les champs input de la page. On récupérait les données à l'aide des librairies FormFactory et BoundForm, on déclarait les variables « pseudo » et « password » en String car tout deux étaient des chaînes de caractères. Ensuite on récupérait la valeur des champs à l'aide de la commande ci-dessous ainsi qu'aux nom donnés aux input dans la page HTML.

```
DynamicForm boundForm = formFactory.form().bindFromRequest();
String pseudo = boundForm.get("pseudo");
String password = boundForm.get("password");

@helper.form(routes.LoginController.valider) {
    @helper.CSRF.formField
    <div class="champsFormulaire"><h2>Pseudo :</h2><input type="text" name="pseudo"></div>
    <div class="champsFormulaire"><h2>Password :</h2><input type="password" name="password"></div>
}

77 public Result deconnection() {
78
79     session().remove("name");
80     session().remove("mdp");
81     flash(SUCCESS,DISCONNECT_MESSAGE);
82     return redirect(routes.LoginController.formulaire());
83
84 }
```

- **la fonction déconnexion** : comme son nom l'indique elle servait tout simplement à déconnecter l'utilisateur lors de l'appuie sur le bouton « LOGOUT ». Pour cela il fallait remettre la variable de session à 0 à l'aide d'un remove.

F Page Récupération

La page récupération permettait à tout utilisateur de récupérer son mot de passe perdu, pour cela il devait renseigner son adresse mail dans un input qui vérifiait en premier temps si l'adresse contenait bien un @ et si elle était présente dans la base de donnée. Le token est généré dans la fonction generateCode(). On retrouvait l'utilisateur à l'aide de l'adresse mail entrée dans l'input. Puis on envoyait un mail à cette adresse avec le token.

```

sendTo = boundForm.get("mail");

if (Users.mailPresents(sendTo)) {
    token = generateCode();
    cip = Users.getCipByMail(sendTo);
    tok = new LostPassword(token, cip);
    tok.save();
    this.sendMail(token, sendTo);
    flash(ConstantsMessagesFlash.SUCCESS, ConstantsMessagesFlash.message(ConstantsMessagesFlash.MAILSEND));
    return redirect(routes.LoginController.loginForm());
} else {
    flash(ConstantsMessagesFlash.ERROR, ConstantsMessagesFlash.message(ConstantsMessagesFlash.MAILNOTFOUND));
    return redirect(routes.PasswordForgotController.sendMailResetForm());
}

```

Une fois ce test effectué, un mail de récupération était envoyé à l'adresse renseigné auparavant, dans ce mail on retrouvait un code de 20 caractères généré aléatoirement par le controller.

```

public String generateCode() {
    String chars = "abcdefghijklmnopqrstuvwxyz1234567890";
    String hash = "";

    for (int nbCaractere = 0; nbCaractere < 20; nbCaractere++) {
        int i = (int) Math.floor(Math.random() * (chars.length() - 1));
        hash += chars.charAt(i);
    }

    return hash;
}

```

Cependant le lien expire au bout d'une durée de cinq minutes c'est une contrainte obligeant l'utilisateur à le faire immédiatement. Pour modifier son mot de passe l'utilisateur était redirigé vers une page où il pouvait rentrer son nouveau mot de passe et le valider afin qu'il soit sauvegardé dans la base de donnée. Pour réaliser cette tâche nous avons eu besoin de différentes bibliothèques supplémentaires.

G Page Projet

La page Projet était accessible à tout le monde. Sur cette page on pouvait observer la liste de tous les projets du laboratoire. L'utilisateur avait la possibilité de cliquer sur un projet, ce qui le redirigeait vers la page Home du projet. Les projets étaient affichés à l'aide d'une boucle for qui parcourait une liste qui récupérait les nom dans la base de donnée.

Cette liste était défini en liste de projet :

```
List<Project> listProjects = Project.find.all();
```

elle récupère tout les projets existant à l'aide de la commande `Project.find.all()` qui fait appelle au modèle `Project` qui va parcourir la base de donnée. Ensuite on parcours cette liste dans la vue à l'aide d'un for :

```
3 @(listProjects : List[Project])
4 @defining(play.core.PlayVersion.current) { version =>
5
6   <h1 id="lprojectspage"></h1>
7   <div class="listProject">
8     @for(project <- listProjects) {
9       <a href=@routes.HomeController.homes(project.getName())>@project.getName()</a>
10    }
11  </div>
12 }
```

à chaque itération, on affichait le nom du projet qui faisait office de lien pour rediriger l'utilisateur vers le projet.

H Page MyProfil

Cette page offrait la possibilité à toute personne connecté de modifier certaines de ses informations renseignées lors de l'enregistrement. Il pouvait notamment modifier son mot de passe, son nom, son prénom, son adresse mail et sa photo ; en revanche il y avait des champs inaccessible pour un Utilisateur comme son statut (User, Admin, Owner) ou encore son rôle à l'université (Étudiant, Professeur, etc). Seul le Super Administrateur pouvait modifier ses champs.

La page pour éditer son profil ressemblait énormément à la page register vue précédemment dans son mode de fonctionnement, elle était constitué de champs à remplir, le mode de fonctionnement est simple, une fois les champs remplis, le bouton « VALIDER » permettait d'enregistrer les données dans des variables qui récupéraient la valeurs de chaque champs grâce aux librairies `BoundForm` et `FormFactory`.

Pour modifier les informations d'un utilisateur il faut dans un premier temps récupérer les informations de cet utilisateur, pour cela nous avons utilisé une variable de session contenant l'id de la personne connecté, à partir de cette variable l'utilisateur était retrouvé dans la base de donnée.

```
141 public Result editMyProfileFormValidation String id {
142
143     Users user = Users.find.byId(id);
```

Ensuite pour changer les données déjà sauvegardées dans son profil, on utilisait comme paramètres les variables précédentes. Enfin on enregistrait la modification du user avec la commande « .save() ».

```
user.setFirstName(boundForm.get("firstname"));
user.setLastName(boundForm.get("lastname"));
user.setMail(boundForm.get("mail"));
user.setResumeEn(boundForm.get("resumeEn"));
user.setResumeFr(boundForm.get("resumeFr"));

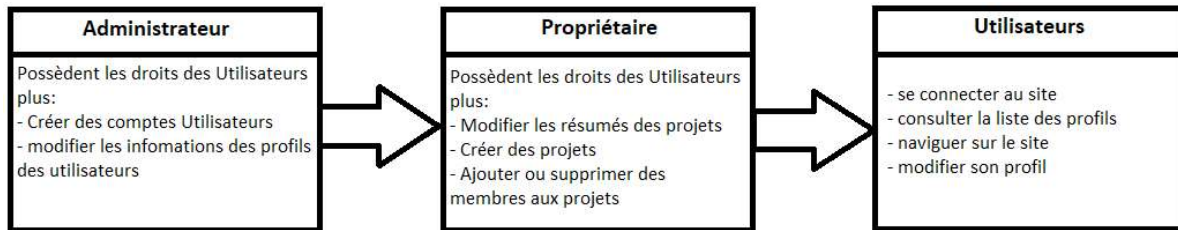
</h2><input type="text" value=@user.getFirstName() name="firstname" />
</h2><input type="text" value=@user.getLastName() name="lastname" />
<input type="mail" value=@user.getMail() name="mail" size="30" />
</h2><input type="password" name="password" size="30" />
</div>
<input type="password" name="verifPassword" size="30" />
```

I Page Gestion Profil

La page Gestion Profile, était disponible pour tout le monde. On y retrouvait la liste de tous les profils inscrit dans la base de données, mais le Super Administrateur et les Propriétaires avaient un bouton qui apparaissait en plus. Ce bouton renvoyait vers le mode édition de profil de n'importe quel utilisateur. Tout comme pour la page projet, il s'agit d'une liste qui est parcouru par une boucle for, seulement cette fois-ci la liste était défini en liste de User.

J Page Création de Compte

Lors de la réunion nous avons décidés de trois statuts différents, les Super Administrateurs, les Propriétaires et les Utilisateurs. Cette page était accessible seulement par le Super Administrateur.



On y retrouvait plusieurs champs à remplir tels que le pseudo qui servait à la connexion, le prénom et le nom, l'adresse mail qui pouvait servir à contacter la personnes ou à la récupération de son mot de passe vu au premier sprint, un mot de passe générer, son rôle au sein du projet (professeur, étudiant, doctorant, etc) et bien sûr son statut (User, Owner, Admin). Pour récupérer toutes ces informations nous avons réalisés un formulaire, une fois les champs remplis et validés, les informations étaient récupérées puis initialisées dans des variables. Ensuite pour créer l'utilisateur on utilisait la fonction User qui était définis dans le modèle User. Chaque variable initialisée correspondait à un paramètre de la fonction. Enfin la fonction save permettait d'enregistrer l'utilisateur dans la base de donnée.

```

42  public Users(String cip, String password, String firstName, String lastName, String
    mail, Status userStatus) {
43
44      this.cip = cip;
45      this.password = password;
46      this.firstName = firstName;
47      this.lastName = lastName;
48      this.mail = mail;
49      this.userStatus = userStatus;
50      this.userStatus = null;
51  }
52
62      newUser = new Users(cip, password, firstName, lastName, mail,
    Status.find.byId(status));
63      newUser.save();

```

Une fois le compte créer, la personne recevait un mail lui demandant de se connecter afin de modifier ses informations personnelles si elles étaient incorrect, de changer son mot de passe et de faire un résumé de sa personne.

K Page Team Active

Sur cette page, on y retrouvait la liste des participants du projet qui étaient encore actifs, c'est à dire qu'ils sont toujours à l'Université de Sherbrooke ou qu'ils participent encore au projet. On utilisait une fois de plus une liste que l'on a parcourus à l'aide d'une boucle for dans la vue.

```
@for(status <- listStatus) {
  <h2>@status.getName()</h2>
  <div class="teamElement">
    @for(membre <- listActive) {
      @if(status==membre.getStatus()) {
        <div class="userTeam">
          <a href=@routes.ProfileController.profile(membre.getCip())><h3>@membre.getFirstName() @membre.getLastName()</h3></a>
          <a href="mailto:@membre.getEmail()">@membre.getEmail()</a>
        </div>
      }
    }
  </div>
}
```

Cette liste de type User, récupérerait tous les utilisateurs qui étaient actif dans le projet. Pour catégoriser l'activité des utilisateurs, nous leurs avons affectés un nombre. Les nouveaux utilisateurs étaient initialisés à la valeur -1, ceux qui étaient actif dans le projet avaient pour valeur 1 et enfin ceux qui étaient inactif étaient initialisés à 0. On a donc parcourus la liste des utilisateurs en vérifiant la valeurs de leurs activités.

L Add Project

Cette page avait pour but de permettre aux propriétaires d'ajouter de nouveau projet. J'ai donc travaillé sur cette page durant le troisième sprint. Pour mener à bien cette mission, j'ai utilisé le modèle Project. Pour la création d'un projet j'avais besoin de deux paramètre, le premier était le nom du projet et ensuite un propriétaire.

```
44 public Project(String name, Owner owner) {
45     this.name=name;
46     this.owners.add(owner);
47 }
```

Pour le nom du projet j'ai utilisé un input qui me renvoyait la chaîne de caractère entrée par le propriétaire, ensuite j'ai initialisé une variable « name » contenant le nom récupéré. Pour le propriétaire, je devais récupérer les informations de l'utilisateur connecté pour vérifié dans un premier temps qu'il s'agissait bien d'un propriétaire. Ensuite je récupère le propriétaire dans une variable « owner ».


```
DynamicForm boundForm = formFactory.form().bindFromRequest();
String projectname = boundForm.get("name");
Project newProject;
ProjectUserParticipation ownerInProject;
Result page;

Owner owner=Owner.getOwnerByCIP(session("name"));
```

Pour la création du projet j'ai fais appelle à la méthode Project avec les deux paramètres récupérés précédemment puis je sauvegarde le projet dans la base de donnée. A la fin de ma fonction j'ai utilisé le modèle « ProjectUserParticipation » qui consistait à modifier l'appartenance au projet, comme le propriétaire faisait partis du projet qu'il avait créé et qu'il est membre actif, j'ai modifié la valeur de 0 à 1.

```
newProject = new Project(projectname, owner);
newProject.save();
ownerInProject=new ProjectUserParticipation(session("name"),projectname);
ownerInProject.inActivity();
owner.addProject(newProject);
owner.save();
```

M Add Team list

Dans cette partie nous avons réalisé la fonctionnalité permettant à un Propriétaire de projet d'ajouter des membres dans son équipe.

Pour la phase d'ajout d'Utilisateur, j'ai utilisé une liste déroulante composé de tous les membres qui faisaient partis de la base de donnée mais qui n'étaient pas présent dans le projet. Pour cela j'ai utilisé la fonction « ProjectUserParticipation » du modèle du même nom. Dans la liste déroulante on retrouvait une liste des cip des utilisateurs, récupérés à l'aide d'une liste. Tout d'abord j'ai récupéré la lise de tous les Utilisateurs ensuite avec une ArrayList, je vérifiais l'appartenance au projet de chaque personne. J'ai initialisé deux listes en plus contenant la liste des personnes active dans le projet et une contenant la liste des personnes inactive.

```
List<Users> listUsersInProject = ProjectUserParticipation.getProjectUsers(Project.find.byId(session("project")));
ArrayList<Users> listUsersNotInProject = this.getNotMembers(listUsersInProject);
List<Users> listActive = ProjectUserParticipation.getProjectUsersActive(Project.find.byId(session("project")));
List<Users> listInactive = ProjectUserParticipation.getProjectUsersInactive(Project.find.byId(session("project")));
```

Ces trois listes étaient ensuite envoyées comme argument dans la page EditTeamProto.

```
page = ok(views.html.templates.TemplateOwner.render("Edit Team", views.html.teams.EditTeamProto.render(listActive, listInactive, listUsersNotInProject)));
```

La page était divisée en trois parties, dans chacune d'elles on y retrouvait une liste d'utilisateur. Je vais seulement parler de l'exemple de la partie concernant les membres actifs, j'ai donc parcouru la liste fournie par le contrôleur en affichant à chaque fois le nom, le prénom, et l'adresse mail de la personne. En-dessous de chaque membre, deux boutons étaient disponibles, comme l'exemple présenté s'agit des membres actifs les deux boutons servaient à passer l'utilisateur dans la partie membre inactif ou supprimer cet utilisateur du projet.

```
@for(membre <- Amembres){
  <div class="members">
    <a href=@routes.ProfileController.profile(membre.getCip()) class="userTeam">@membre.getFirstName() @membre.getLastName()</a>
    <div class="LbuttonTeam">
      <div class="divButtonTeam">
        <a href=@routes.TeamController.protoPostDesactivateUserTeam(membre.getCip()) class="button_Team">
          <div class="minus_icon">
            </div>
        </a>
        <p id="bdeactivate"></p>
      </div>
      <div class="divButtonTeam">
        <a href=@routes.TeamController.protoDeleteUserTeam(membre.getCip()) class="button_Team">
          <div class="trash_icon"></div>
        </a>
        <p id="bremove"></p>
      </div>
    </div>
  </div>
}
```

N Traduction Anglais/Français

Lors de notre réunion, Ruben nous a demandé d'intégrer la traduction du site Web en anglais et en français. Pour répondre aux demandes nous avons ajouté deux boutons sur le template permettant de switcher entre la langue française et anglaise.

Pour traduire la totalité du site nous avons créé un dictionnaire en javascript composée de deux fonctions une anglais() et une français(), ces fonctions sont composées notamment de tests. Prenons l'exemple de bouton sur le site :

Français :

```
381 //BUTTONS
382 if(document.getElementById("baddproject")!=null)
383 {
384     document.getElementById("baddproject").innerHTML = "Add Project";
385 }
386
387 if(document.getElementById("bedit")!=null)
388 {
389     document.getElementById("bedit").innerHTML = "Edit";
390 }
```

Anglais :

```
381 //BUTTONS
382 if(document.getElementById("baddproject")!=null)
383 {
384     document.getElementById("baddproject").innerHTML = "Add Project";
385 }
386
387 if(document.getElementById("bedit")!=null)
388 {
389     document.getElementById("bedit").innerHTML = "Edit";
390 }
```

```
<button type="submit" id="baddproject"></button>
```

On peut voir ci-dessus que dans un premier temps on récupérait l'élément, ici il s'agit de nom de boutons. L'id du bouton a pour nom « baddproject », on récupérait donc cet id. Ensuite selon la langue choisie par l'utilisateur, soit on faisait appel à la fonction `english` soit la fonction `français`, enfin on lui attribuait une chaîne de caractère qui était affichée sur le bouton. Tout le texte du site que nous avons développé possédait un id qui permettait de traduire le site dans deux langues différentes, une fois le système mis en place rien ne nous empêchait d'ajouter de nouvelles langues.

O Autre Page

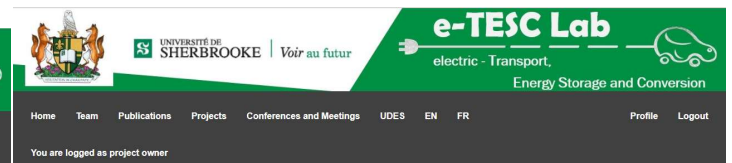
Lors d'une réunion avec notre tuteur au premier sprint, nous avons décidé que dans premier temps seul les pages Home, Login, Récupération de mot de passe, le header et le footer étaient importantes, c'est pour ça que nous avons juste créer les pages Edit my CV, Research, et Contact. Ces pages devaient être développées dans la suite du projet , seulement nous avons modifier le but final du projet ces pages ce sont donc transformer par la suite.



Contact

Université de Sherbrooke :
Campus Principal : 819 821-8000

This project was realized by Emilien Cosnier, Vincent Le Page, Julien Beuve and Loïc Travaillé



Publications

Search:

This project was realized by Emilien Cosnier, Vincent Le Page, Julien Beuve and Loïc Travaillé



Edit my CV

This project was realized by Emilien Cosnier, Vincent Le Page, Julien Beuve and Loïc Travaillé

P Les tests

Chaque fonctionnalité du site devait être testée pour que lorsque que l'un des développeurs modifie un contrôleur ou un modèle, il puisse vérifier que ses modifications n'avaient pas créé de bug. La plupart des tests se ressemblaient c'est pourquoi je ne présenterai que celui de la page Home.

Ce test avait pour but de tester le bon affichage des pages Home selon si on est un Utilisateur ou bien un Propriétaire, ou bien le changement de page comme par exemple pour aller sur la page d'édition .

Dans un premier temps j'ai initialisé dans un before tout ce dont j'avais besoin pour les tests c'est à dire un utilisateur, un propriétaire ,un projet et une fausse application qui faisait office de simulation de la vraie application.

Ensuite les tests commençaient, ci-dessous vous pouvez voir un test qui avait pour but de vérifier que la page Home utilisait bien la bonne route pour s'afficher :

```
@Test
public void renderHome(){

    request = new Http.RequestBuilder().method("GET").uri(routes.HomeController.home().url());
    Result result = route(app,request);
    assertEquals("This is not the page", "/", request.uri());
    assertEquals(OK, result.status());

}

ownerTest = new Owner(user0Test);
ownerTest.save();
pTest= new Project(PROJECT, "", "", ownerTest);
pTest.save();

app = fakeApplication();
request = new Http.RequestBuilder();
start(app);

}
```

La variable « request » faisait appel à la bonne route à utiliser pour l'affichage de la page, puis on demandait à l'application d'utiliser cette route. Enfin à l'aide de la commande « assertEquals() » j'ai vérifié si la route affichée, ici « / » correspondait à la route initialisée dans la requête. Le deuxième test vérifiait si la page chargée avait le même statut que OK ou 202. Le but de la réponse 202 est de permettre à un serveur d'accepter une requête pour un autre processus.

```
@Test
public void renderHome(){

    request = new Http.RequestBuilder().method("GET").uri(routes.HomeController.home().url());
    Result result = route(app,request);
    assertEquals("This is not the page", "/", request.uri());
    assertEquals(OK, result.status());

}
```

Ensuite j'ai testé le changement de page en passant de la page Home à sa page d'édition.

```
@Test
public void testToGoEditHomeNotAdmin() {

    request = new Http.RequestBuilder().method("GET").uri(routes.HomeController.editHome().url());
    Result result = route(app,request);
    assertEquals("This is not the page", "/", result.redirectLocation().get());
    assertEquals(303, result.status());

}
```

Tout comme dans le test présenté avant, j'ai initialisé une requête avec la route de la page EditHome. Seulement cette fois ci mon test consistait à vérifier si un Utilisateur essayait d'aller sur cette page il devait être redirigé vers la page Home car il n'avait pas les droits. C'est pourquoi cette fois-ci j'ai utilisé la réponse 303 qui signifie que la page chargée a changée, ici l'Utilisateur est passé de la page EditHome à la page Home.

Conclusion

A Conclusion technique

Arrivé à la fin des 11 semaines de stage, nous avons réussi à réaliser un site fonctionnel. Dans l'ensemble le résultat correspond aux attentes de notre tuteur Monsieur Gozalez-Rubio Ruben. Bien sûr la phase d'apprentissage a été très importante durant ce stage pour me permettre d'avancer. Nous avons rencontré quelques problèmes dans notre développement, notamment liés à la base de données où seuls deux personnes pouvaient se connecter en même temps, les deux dernières semaines nous avons eu beaucoup de mal à nous connecter et donc cela nous empêchait de travailler efficacement.

Notre site de gestion de projet peut permettre aux différentes personnes de créer puis de gérer leurs projets en y ajoutant des membres, en modifiant les résumés, ou encore modifier son profil. Le système de statut permet de classer toutes les personnes inscrites sur le site Web.

Le résultat obtenu reste néanmoins une ébauche, avec plus de temps certaines fonctionnalités auraient pu être ajoutées ou optimiser celles déjà existantes comme par exemple changer la bannière en fonction du projet actif, l'ajout de photo sur le site pour illustrer le projet était en cours de développement lors de la fin du stage mais n'était pas encore au point, mettre en place un système de recherche pour trouver une publication à l'aide de critères tels que l'auteur ou encore le titre. Pour l'avenir c'est l'Université qui décidera de la réalisation du projet.

B Conclusion Personnelle

Ce stage au Québec m'a permis de découvrir des méthodes de travail différents que celles enseignées à l'IUT, notamment avec la méthode Scrum ou l'architecture MVC. J'avais choisis ce stage pour approfondir mes connaissances dans le domaine de l'informatique, cet objectif a été rempli. Au début du stage je ne connaissais que des bases en Java et en HTML mais à force de recherche et persévérance j'ai su m'adapter pour rattraper mon retard. Le fait de travailler dans une équipe de développement m'a permis de travailler en autonomie tout en devant communiquer avec mes collaborateurs.

Ce stage m'a permis d'acquérir de l'expérience et des connaissances qui vont m'être utiles dans mon projet futur. Bien que le stage se soit très bien passé, je pense pas être fait pour rester derrière un bureau toute la journée mais plutôt d'allier l'informatique avec de la pratique, c'est pourquoi pour l'année prochaine je suis à la recherche d'une alternance dans le domaine de la maintenance d'automate.

Je vous remercie d'avoir lu mon rapport de stage.

Annexe

Résumé Clean Code

Pour un programmeur, il est important de bien structurer son code afin de simplifier sa compréhension pour lui ainsi que pour toutes les personnes amenées à lire ce code. Nous allons voir les différentes étapes pour arriver à un code «propre» d'après le livre «Clean Code».

Tout d'abord qu'est-ce qu'un mauvais code. Un mauvais code c'est lorsque que les fichiers ne sont pas organisés correctement ce qui empêche une lecture rapide et efficace du code, cela entraîne une perte de temps importante qui peut empêcher de finir le projet dans les temps. Bien souvent les programmeurs veulent aller trop vite, si bien que leurs codes sont en désordre et ils rajoutent de plus en plus de fichiers.

En résumé pour éviter d'avoir un mauvais code, il faut prendre le temps de réfléchir à son code, à son architecture afin de pas se perdre lors de la programmation, sinon on ne pourra pas réaliser le projet avant la fin du temps. Rien ne sert de courir il faut partir à point.

Pour réaliser un code facile à comprendre, il faut d'abord donner des noms aux classes, aux fonctions et aux variables qui ont du sens, par exemple si l'on crée une fonction nous permettant de simuler une boîte noire d'un avion on ne doit l'appeler «btn» ou encore «chocolat». Les noms doivent facilement être prononçables pour favoriser la communication entre les membres d'une équipe de développement.

Chaque partie de code doit respecter certaines contraintes afin de maximiser la lisibilité du programme. Pour les Classes, elles ne doivent pas dépasser 500 lignes, comme chaque classe correspond à une entité de ce monde comme par exemple une voiture. Dans le cas des Fonctions, ses méthodes doivent être expressives et de petite taille il ne faut pas dépasser les 20 lignes. Chaque fonction ne traite qu'une seule fonctionnalité, si l'on reste sur l'exemple de la voiture une fonction peut s'occuper par exemple du système de freinage et uniquement de celui-ci. Enfin les fonctions ne doivent pas avoir plus de deux paramètres de fonction, le mieux est de ne pas en avoir du tout.

Souvent on nous dit que pour qu'un programme soit facile à comprendre il faut lui ajouter des commentaires, or un bon code est censé se suffire à lui-même grâce aux noms des variables explicites et d'une architecture logique dans notre programme.

Pour mener à bien un projet, il est important de réaliser des tests tout au long de la programmation pour s'assurer du bon fonctionnement des parties déjà conçues. Cependant il ne faut pas créer plus de tests que nécessaires.

Dans chaque programme, il y a des exceptions qui apparaissent suites à des entrées anormales où les appareils peuvent échouer. Le but d'une exception est de permettre au code de continuer à faire ce pourquoi il a été créé. Cependant ces exceptions doivent être claires, dans certains codes la gestion des erreurs est tellement dispersée qu'il est impossible de voir d'où provient l'erreur. L'utilisation de la commande `try{} catch(){} finally{}`, permet de clarifier le code et de séparer la gestion des erreurs et l'arrêt du périphérique. Grâce à cette séparation on a la possibilité de lire et comprendre indépendamment chaque partie. Lorsque l'on essaye d'accéder à un fichier inexistant, le test ne génère pas d'exception ; cependant si l'on cherche un fichier invalide, on obtient une exception. La signature de chaque méthode énumérerait toutes les exceptions qu'il pourrait transmettre à son appelant. De plus, ces exceptions faisaient partie du type de la méthode. À l'époque, nous pensions que les exceptions vérifiées étaient une excellente idée, ils peuvent donner un certain avantage. Cependant, il est clair maintenant qu'ils ne sont pas nécessaires à la production de logiciel robuste. Si l'une des fonctions d'un niveau très bas est modifiée de telle sorte qu'elle doit lancer une exception, on doit ajouter une clause `throws` à cette dernière. Mais ça signifie que chaque fonction qui appelle notre fonction doit ajouter une clause `throws`. À cause de cela l'encapsulation est interrompue, car chaque fonction doit connaître les détails de cette exception de bas niveau.

Une partie est consacrée aux tests unitaires, les tests occupent une place importante dans un projet. C'est une procédure permettant de vérifier qu'il répond aux spécifications fonctionnelles et qu'il fonctionne correctement en toutes circonstances, par exemple l'affichage d'une page HTML ou le changement du mot de passe d'un utilisateur. Les tests doivent être petits avec seulement une fonctionnalité par test, ils doivent répondre aux critères F.I.R.S.T qui signifie :

- Fast : ils doivent être rapides pour les lancer rapidement
- Independent : les tests ne doivent pas être dépendants d'autres tests
- Repeatable : le programmeur doit pouvoir recommencer les tests autant de fois qu'il le désire
- Self-validing : chaque test retourne soit la valeur «true» soit «false»
- Thorough and Timely : les tests doivent couvrir la totalité des scénarios d'utilisation

En conclusion, bien que «Clean Code» soit paru il y a maintenant dix ans, les points abordés par l’auteur dans son ouvrage restent véridiques. «Clean Code» fait le point sur les bonnes pratiques de développement permettant d’aboutir à un code source de qualité, clair, bien construit et organisé, testé et par conséquent maintenable et évolutif. Le livre donne au lecteur les clés pour produire du code propre et pour nettoyer du code de moindre qualité. Ce livre s’adresse aux développeurs mais également aux ingénieurs, analystes et concepteurs logiciels, aux chefs de projet et à tous les agilistes.

Projet

Voici un lien pour récupérer le projet sur mon drive, en revanche sans la base de donnée vous ne pourrez pas lancer le projet :

<https://drive.google.com/drive/folders/1ovK4rLv2T76S-hcRRaGsPp4gr-7iLfuS>