

Aymeric Moreau

STAGE : Stock2Com



Remerciement

Je remercie Monsieur Hys le président de l'entreprise pour m'avoir accepté dans son entreprise. Je remercie julien mon maître de stage pour nous avoir aiguillé lorsqu'on en avait besoin. Je remercie mon collègue Mickael pour nous avoir aidé lorsqu'on le demandait. Je remercie ma professeure référente Madame Delouche pour sont suivis lors du stage.

Sommaire

L'entreprise : Stock2Com.....	4
Environnement Matériel.....	4
Contexte Professionnel.....	4
Mission Confierée.....	4
Gestion de Projet.....	5
Rétrospective Hebdomadaire.....	5
Semaine 1.....	6
Semaine 2.....	7
Semaine 3.....	23
Semaine 4.....	30
Semaine 5.....	37
Semaine 6.....	42
Semaine 7.....	60
Bilan du stage.....	67
Les objectifs :.....	67
Ce que m'a apporté mon stage :.....	67
Conclusion.....	68

L'entreprise : Stock2Com

Stock2Com est une entreprise spécialisée dans le développement Web. Elle se situe à 24, rue des Cordeliers Tours. Son gérant et créateur est René Hys.

L'entreprise a par exemple développé le CMS (site permettant de créer des sites internet facilement) « STOCK2COM-AV » qui est un CMS Dédié aux agences de voyage et « MAP RAPTOR ® » qui est une application de création et de gestion de cartographies interactives. L'entreprise est composée de René Hys sont gérants, Yves Mbonigaba un Administrateur Système, Mickaël Maronneau un développeur, Julien un développeur et Sakula une secrétaire.

Environnement Matériel

Nous Stagiaire nous sommes en BYOD nous devons amener notre propre matériel.

Les employer eux sont sur le matériel de l'entreprise.

Contexte Professionnel

Nous sommes des employer de l'entreprise Stock2Com et nous devons nous occuper de la refonte graphique de certaine page du CMS de Stock2Com.

Mission Confiee

1)

Notre première mission consiste à nous familiariser avec le langage de programmation Ruby et le framework Ruby On Rails utilisés dans leur application.

Pour atteindre cet objectif, nous devons suivre des formations disponibles sur le site Grafikart. J'ai déjà suivi la formation « **Apprendre Ruby** » (<https://grafikart.fr/formations/ruby>) juste avant le début du stage.

Nous avons également la formation « **Apprendre Ruby on Rails** » (<https://grafikart.fr/formations/ruby-on-rails>) à compléter.

Bien que nous n'ayons pas de délai strict, nous savons qu'en général, les stagiaires consacrent environ 10 jours à la réalisation de l'ensemble des formations. Cette estimation nous donne une idée du planning, même si elle peut varier en fonction de nos progrès individuels.

2)

Notre 2ème mission est de nous occuper de la mise à jour graphique de certaine page du CMS. Je m'occupe de la page de gestion des utilisateurs.

3) La troisième mission consiste à nous occuper de la mise à jour graphique ainsi que du regroupement de quatre pages en une seule. Nous allons fusionner les pages 'Mentions légales', 'Liens copyright', 'Mode de paiement' et 'Politique de confidentialité' en une seule page appelée 'Informations légales & paiement'. C'est notre projet à tous les deux, avec mon collègue. Il nous revient de nous répartir les tâches.

Gestion de Projet

Pour la gestion de projet, l'entreprise ne nous a rien imposé. Nous avons la liberté de nous organiser selon nos préférences. Mon collègue stagiaire et moi avons décidé d'utiliser Trello.

Pour le 1^{er} projet les pages nous sont attribués par l'entreprise : j'ai une page à réaliser, et mon collègue en a une autre.

La validation de notre travail se fait après que nous avons terminé notre projet.

Pour le 2^{ème} projet nous avons décidé de faire un diagramme de Grantt. C'est à nous de décider qui fait quoi dans la page et avec quel délai.

Rétrospective Hebdomadaire

Semaine 1

J'ai passé toute la première semaine à me former sur la formation qui me restait à accomplir, "Apprendre Ruby on Rails" de Grafikart. J'avais déjà commencé cette formation avant le stage, mais je n'avais pu que procéder à l'installation du framework Ruby on Rails sur ma machine.

Ruby on Rails est un framework de développement web côté serveur open-source basé sur le langage de programmation Ruby. Il repose sur le modèle MVC, un motif d'architecture logicielle qui sépare le projet en trois modules principaux : le Modèle (ce qui est lié aux données et à la base de données), la Vue (l'interface graphique) et le Contrôleur (contenant la logique de l'application).

Dans cette formation, on nous apprend à prendre en main le framework Ruby on Rails. Les sujets abordés comprennent :

- Les migrations : Avec Ruby on Rails, il est possible de créer, à l'aide d'une simple commande, un script Ruby qui modifie la base de données. Cela s'avère utile lors de la transmission du projet à quelqu'un d'autre, car il lui suffira d'exécuter les migrations pour obtenir l'architecture de la base de données.
- Les 3 modules MVC : Le modèle (lié aux données et à la base de données), la vue (interface graphique) et le contrôleur (contenant la logique de l'application).
- Le CRUD : Il englobe la persistance des données, avec les opérations Create, Read, Update et Delete.
- Les filtres : Ils permettent de filtrer les données à divers stades du traitement.
- Les variables de session : Utilisées pour stocker des informations spécifiques à une session utilisateur.
- Les cookies : Des petits fichiers texte stockés sur l'ordinateur du client, utilisés pour suivre les informations de l'utilisateur.
- Les Callbacks : Ce sont des méthodes appelées à certains moments du cycle dans les modèles.
- Les scopes : Ils permettent dans le modèle d'envoyer des données plus précises. Par exemple, si on a un modèle Post, on peut ajouter dans le modèle : scope :offline, -> {where(online: false)}. Cela permet dans le contrôleur de récupérer uniquement les articles hors ligne, remplaçant ainsi @posts = Post.where(online: false).all par @posts = Post.offline.all.
- Les associations entre modèle
- Les partials : ce sont des fichiers vues séparés que l'on peut inclure dans plusieurs fichiers vues. Par exemple, le formulaire de création et de modification, souvent identique, peut être placé dans un partial et inclus ensuite dans la vue.
- Le layout : c'est la page sur laquelle toutes les vues vont s'inclure.
- Les assets : c'est l'endroit qui regroupe le CSS, les images et les fichiers JavaScript.
- L'envoi de mails.
- Les gems Devise et OmniAuth : elles facilitent grandement la gestion des utilisateurs. Elle permet à l'aide d'une simple commande de générer automatique le système d'authentification donc la création de compte, la connexion, l'envoie de mail de confirmation, la réinitialisation de mot de passe, ect...
- La gem Cancancan : elle simplifie la gestion des droits des utilisateurs de l'application. Elle permet simplement de créer un droit (par exemple lire les données venant de modèle post) et d'affecter ce droit ou non à un utilisateur.

- La gem SimpleForm : elle simplifie grandement la création de formulaires. Avec cette gem on peut créer par exemple un champ e-mail avec cette simple ligne : `<%= f.input :email %>` Il va générer lui-même le label et l'input lors de l'affichage dans le navigateur.
- Les gems Carrierwave et Shrine : elles facilitent l'upload et la gestion des fichiers.
- Le fragment caching : activer la mise en cache sur certaines parties du site web pour optimiser sa vitesse.
- La gem RSpec : elle permet de réaliser des tests unitaires.

J'ai pu terminer la formation en une semaine. La plupart des problèmes que j'ai rencontrés sont dus au fait que la formation est conçue pour la version Ruby on Rails 5.0, alors que j'étais sur la 7.0. Cependant, la plupart des solutions étaient disponibles dans les commentaires de la formation.

Semaine 2

Maintenant que la formation est terminée, nous pouvons passer au projet. Nous avons débuté la semaine par une réunion en visioconférence avec Julien, le développeur chargé de préparer notre projet.

Mon objectif est de m'occuper de la refonte graphique de la page de gestion des utilisateurs dans la partie gestion du CMS de Stock2Com. Voici à quoi ressemble la page actuellement :

Aymeric Moreau

The screenshot shows the Stock2Com administration interface. The left sidebar contains a navigation menu with sections like 'Gestion', 'Accueil', 'Devis / Réervations', 'Messages / Avis', 'Produits', 'Site > Configuration', 'Utilisateurs' (selected), 'Coordonnées', 'Mentions légales', 'Politique de confidentialité', 'Liens Copyright', 'Horaires de rappel', 'Notifications mail', 'Modes de paiement', 'Site > Apparence', 'Site > Contenus', 'Site > Blog', 'Site > Outils', 'Newsletters', 'Boîtes mail', and 'Journal des événements'. The main content area is titled 'Gestion des utilisateurs' and shows a form to 'Créer un utilisateur' (Create user) with fields for 'Nom contenant' (Containing name) and 'Rôles' (Roles). Below the form is a table listing existing users:

Nom / Prénom	Email	Rôle	Groupe	Actions
Stock2Com Admin	admin@stock2com.com	Administrateur	-	
Stock2Com Demo	demo@stock2com.com	Administrateur	-	
Stock2Com Manager	manager@stock2com.com	Manager	-	
Stock2Com User	user@stock2com.com	Utilisateur Gestion	-	

At the bottom of the sidebar, there is contact information for Stock2Com: 'stock2com developpements', '06 63 27 57 05', and 'support@stock2com.com'.

Et voici la version que je dois créer :

Aymeric Moreau

Nom / Prénom	Email	Rôle	Groupe			
HYS René	rene@stock2com.com	Utilisateur gestion	Groupe Managersd			
HYS René	rene@stock2com.com	Utilisateur gestion	Groupe Managersd			
HYS René	rene@stock2com.com	Utilisateur gestion	Groupe Managersd			

GESTION DES GROUPES / RESTRICTIONS D'ACCÈS

Groupes	Restrictions	Modifier	Voir	Supprimer
Clients Pro				
Managers				
Utilisateurs Gestion				

Modale «Ajouter un utilisateur»

* Prénom: _____ * Nom: _____

Email: _____

Mot de passe: _____ Confirmation du mot de passe: _____

* Rôles: Manager Groupe: _____

ENREGISTRER **FERMER**

Modale «Restrictions»

Autoriser tout - Bloquer tout

Authentification

- Gestion des droits (Valeur par défaut)
- Gestion des groupes (Valeur par défaut)
- Gestion des utilisateurs (Valeur par défaut)
- Modification de son profil utilisateur (Valeur par défaut)
- Voir les profils utilisateurs de son groupe (Valeur par défaut)

Journal d'événements

- Gestion de l'historique (Valeur par défaut)

ENREGISTRER **FERMER**

Je n'ai pas reçu de limite de temps, excepté la durée de mon stage. L'entreprise a spécialement mis en place un serveur de test pour les stagiaires. Ainsi, nous ne coderons pas directement sur le site web en ligne. Tout ce que nous ferons n'aura aucun impact sur l'infrastructure de l'entreprise, et il n'y a aucun risque de causer des problèmes au site. Nous avons la liberté d'expérimenter tout ce que nous souhaiterons.

Je n'ai pas d'outils imposés ; il m'appartient de choisir ceux que je souhaite utiliser pour travailler. Pour communiquer avec le serveur en SSH, j'ai opté pour le logiciel MobaXterm. C'est une boîte à outils réseau qui permet, entre autres, de créer des sessions SSH, Telnet, FTP, SFTP, RDP, VNC, RSH, ou encore Xdmcp. J'ai choisi ce logiciel car il offre la possibilité, lors de la connexion en SSH à un serveur, d'avoir accès à l'arborescence des fichiers avec la capacité de les modifier via l'éditeur de code que je souhaite. Pour la programmation, j'ai décidé d'utiliser Visual Studio Code, l'éditeur de code avec lequel j'ai suivi ma formation.

Maintenant que tout est en place, nous avons pu commencer le projet. Nous avons choisi pour la gestion de projet d'utiliser des tableau Trello un chacun. La conception du tableau a été plutot compliquer car il fallait comprendre le code et la structure des page pour savoir comment séparé les tache. Voici a quoi ressemble le tableau :



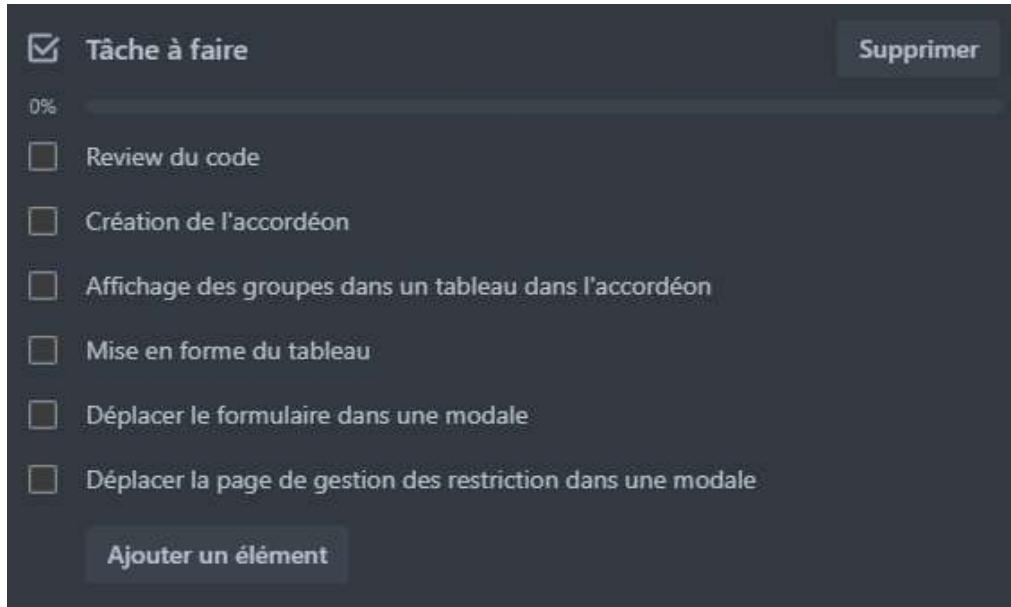
Carte utilisateur :

The screenshot shows a Trello card titled "Tâche à faire". The card has a progress bar at 0% completion. Below the title, there is a list of tasks with checkboxes:

- Review du code
- Affichage des utilisateurs dans un tableau
- Mise en forme du tableau
- Déplacement du formulaire dans une modale

At the bottom of the card, there is a "Supprimer" button and a "Ajouter un élément" button.

Carte groupe :



Carte Assemblage de la page :



J'espère cette 1ere semaine que je vais avoir le temps de m'occuper des fonctionnalités principales de la page, la gestion des utilisateurs et des groupes mais pas les restrictions car c'est ce qui est le plus flou pour moi pour l'instant.

La premiere étape est donc lire le code et essayer de le comprendre. À peine avons-nous commencé à accéder au site web pour voir à quoi il ressemble, qu'un problème survient :

Sass::SyntaxError in Home#index

Showing /var/s2cstag1/noyau/templates/home_full_screen/app/themes/home_full_screen/views/shared/_head.html.erb where line #10 raised:

```
File to import not found or unreadable: jquery.mCustomScrollbar.
Load paths:
/var/s2cstag1/noyau/apps/demo/app/assets/images
/var/s2cstag1/noyau/apps/demo/app/assets/javascripts
/var/s2cstag1/noyau/apps/demo/app/assets/stylesheets
/var/s2cstag1/noyau/apps/demo/vendor/assets/javascripts
/var/s2cstag1/noyau/apps/demo/vendor/assets/stylesheets
/var/s2cstag1/noyau/modules/newsletters/app/assets/images
/var/s2cstag1/noyau/modules/newsletters/app/assets/javascripts
/var/s2cstag1/noyau/modules/newsletters/app/assets/stylesheets
/var/s2cstag1/noyau/modules/reservations/app/assets/images
/var/s2cstag1/noyau/modules/reservations/app/assets/javascripts
/var/s2cstag1/noyau/modules/reservations/app/assets/stylesheets
/home/webuser/.rbenv/versions/2.4.2/lib/ruby/gems/2.4.0/gems/jquery-validation-rails-1.19.0/vendor/assets/javascripts
/var/s2cstag1/noyau/modules/tracking/app/assets/images
/var/s2cstag1/noyau/modules/tracking/app/assets/javascripts
/var/s2cstag1/noyau/modules/tracking/app/assets/stylesheets
```

Nos collègues se sont chargés de régler le problème, qui était lié à un problème de droits au niveau d'une gemme.

Ainsi, nous pouvons entamer la lecture du code.

Mon premier objectif était de modifier le tableau des utilisateurs. Pour cela, j'ai entrepris une recherche sur le site web pour repérer des éléments similaires à ce tableau. J'en ai identifié un sur la page de gestion des produits, dont la structure était la suivante :

Catégorie	Etat	Référence	Fournisseur	Nom du produit	Actif le	Inactif le				
Indiv			Votre production	Produit Test						
Indiv			Votre production	VOL + BAY GARDEN HOTEL*** 3J/2N + ...						
Indiv			Votre production	Test prod-test copie						
Indiv			Votre production	VOL + BAY GARDEN HOTEL*** 3J/2N + ...						

Ce tableau m'a fourni la base nécessaire, comprenant déjà le tableau en lui-même ainsi que certains boutons à ajuster pour les faire correspondre à mon modèle. Pour le moment, je vais configurer le tableau directement dans le fichier `index.html.erb`, sans recourir à une partial je n'en ai pas besoin pour l'instatn. Voici les modifications apportées au tableau :

- **les colonnes** : Les seules colonnes dont j'aurai besoin sont le nom/prénom, l'email, le rôle et les groupes. Voici le code :

```
<thead>
  <tr>
    <th>
      | <%= ::Authentification::User.human_attribute_name(:name) %> / <%= ::Authentification::User.human_attribute_name(:lastname) %>
    </th>
    <th class="">
      | <%= ::Authentification::User.human_attribute_name(:email) %>
    </th>
    <th class="">
      | <%= ::Authentification::User.human_attribute_name(:roles) %>
    </th>
      | | | <th class="text-center">
        | <%= ::Authentification::User.human_attribute_name(:group) %>
      </th>
    <th>
    </th>
  </tr>
</thead>
```

La ligne `::Authentification::User.human_attribute_name(:name)` est utilisée pour extraire, à partir d'un fichier de traduction `.yml`, la valeur correspondant à l'identifiant. Ces fichiers de traduction servent à mettre à disposition des termes dans différentes langues. La valeur renournée dépend de la langue paramétrée dans le navigateur : si c'est l'anglais, la ligne renverra "name", si c'est le français, elle renverra "prénom". Cette approche simplifiée facilite la gestion des langues.

Cependant, il existe une autre méthode pour afficher les valeurs du fichier de traduction en utilisant `t('activerecord.attributes.authentification/user.name')`. Bien que cette commande accomplisse la même tâche, notre collègue recommande la première méthode, car Ruby on Rails se charge automatiquement de récupérer la valeur. Dans la commande `t()`, il est nécessaire de fournir manuellement le chemin complet permettant d'accéder au terme souhaité dans le fichier de traduction. Voici à quoi ressemble un fichier traduction :

```
---
fr:
  activerecord:
    attributes:
      authentification/user:
        name: Prénom
        lastname: Nom
        email: Email
        password: Mot de passe
        password_confirmation: Confirmation du mot de passe
        role: Rôle
        roles: Rôles
        group: Groupe
        roles_symbols:
          admin: Administrateur
          manager: Manager
          user: Utilisateur Gestion
          visitor: Visiteur Connecté
      authentification/group:
        name: Nom
        users: Utilisateurs associés
      user:
        name: Prénom
        lastname: Nom
```

En haut, nous avons la langue, suivie de différentes clés pour organiser les valeurs.

Le problème qui se pose est que si nous modifions le nom d'une clé pour une raison quelconque, par exemple 'authentification/user', il faudra effectuer la modification partout où une valeur de cette clé est utilisée. En utilisant `::Authentification::User.human_attribute_name(:name)`, il suffit de spécifier le nom du module, du modèle et de la clé souhaitée. C'est la fonction 'human_attribute_name' qui se charge automatiquement de récupérer cette information. La commande `t()` est réservée aux valeurs présentes dans le fichier de traduction global, n'étant pas spécifiquement liées à un modèle ou à un module.

- Affichage des utilisateurs :

Je réutilise le code déjà conçu pour la boucle, incluant le nom/prénom, l'e-mail, le rôle. La modification nécessaire concerne les boutons.

Voici la 1^{er} version du bouton "Modifier" :



```
<%= link_to edit_gestion_user_path(user), class: " btn btn-rectangular btn-success", data: {"toggle" => "tooltip", "placement" => "top"}, title: ::Authentification::User.human_attribute_name(:edit) do %>
  <%= icon('pencil') %>
  <%= t('global.edit') %>
<% end %>
```

Il s'agit d'un lien auquel sont attribuées des propriétés Bootstrap de bouton.

edit_gestion_user_path(user) représente la route qui donne accès au formulaire d'édition. data: {"toggle" => "tooltip", "placement" => "top"} indique l'ouverture d'une info-bulle et spécifie son emplacement. title: ::Authentification::User.human_attribute_name(:edit) attribue un titre à l'info-bulle. Ensuite, <%= icon('pencil') %> et <%= t('global.edit') %> définissent le contenu du bouton, à savoir une icône de crayon et le libellé correspondant à la clé "edit" dans le fichier de traduction global.

Cependant, le problème provient de la manière dont les attributs Bootstrap sont actuellement définis. J'ai géré le style manuellement avec les outils Bootstrap, mais dans le projet, l'uniformité est essentielle. Pour remédier à cela, des classes ont été créées spécialement pour les boutons "Modifier", telles que btn-edit, définies dans le fichier SCSS suivant :

```
.btn.btn-edit{
  color: #fff;
  background-color: $action_bo;
  border-radius: 0;
  font-size: 18px;
  height: 36px;
  &:before{
    content: '\f040'; // icone fa-pencil
    font: normal normal normal 14px/1 FontAwesome;
    font-size: 18px;
  }
}
```

Il est recommandé d'utiliser btn-edit au lieu de btn-rectangular btn-success pour garantir la cohérence visuelle avec d'autres boutons de modification. Cette classe configure directement la couleur du bouton, la police du texte, l'icône, etc

code finale du bouton :

```
<%= link_to edit_gestion_user_path(user), class: "btn btn-edit", data: {"toggle" =>
"tooltip", "placement" => "top"}, title: ::Authentification::User.human_attribute_name(:edit) do %>
<%= t('global.edit') %>
<% end %>
```

-Dernier changement :

J'ai dû retirer des attributs du tableau table-striped table-bordered car ils ne correspondent pas au modèle. Ces deux classes ajoutent des bordures verticales aux valeurs et modifient la couleur d'une ligne sur deux.

Rendu final du tableau :

Prénom / Nom	Email	Rôles	Groupe	Actions
Stock2Com Admin	admin@stock2com.com	Administrateur	-	
Stock2Com Demo	demo@stock2com.com	Administrateur	-	
Stock2Com Manager	manager@stock2com.com	Manager	-	
Stock2Com User	user@stock2com.com	Utilisateur Gestion	-	

Après avoir terminé le tableau, j'ai commencé à examiner la modification de la création d'utilisateurs.

J'ai débuté par la modification du bouton, en récupérant celui présent sur la page de gestion des produits. Ce bouton est en réalité un fichier distinct appelé '_add_btn' et est invoqué à l'aide d'un 'render'.

Par conséquent, j'ai dû créer un fichier '_add_btn.html.erb' dans le dossier des vues pour les utilisateurs, puis copier le code du bouton. Les seules choses à changer étaient le lien, pour qu'il pointe vers les utilisateurs plutôt que vers les produits, le texte affiché sur le bouton, et l'ajout de la classe 'ajax-modal'. Cette classe indique au bouton qu'il ouvrira une modale. J'ai également ajouté cette classe pour le bouton de modification.

```
<div class="row">
  <div class="col-xs-12 col-sm-12 col-md-2 col-md-offset-10">
    <div class="panel panel-green panel-no-border">
      <div class="panel-heading">
        <div class="row">
          <div class="col-xs-12 text-center">
            <%= link_to new_gestion_user_path , class: 'ajax-modal text-decoration-none' do %>
              <%= icon "plus fa-3x" %>
              <h3 class="no-vertical-margin">
                <%= ::Authentification::User.human_attribute_name(:add)%>
              </h3>
            <% end %>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
```

Ensuite on l'injecte sur la vue index :

```
<%= render partial: 'authentification/gestion/users/add_btn' %>
```

Rendu finale du bouton :



Ensuite, je suis passé au formulaire. J'ai commencé par récupérer une modal déjà créée, celle de la création d'un produit. J'ai utilisé la structure de la modal et j'ai intégré dans le corps le formulaire que j'avais déjà à ma disposition.

La modal :

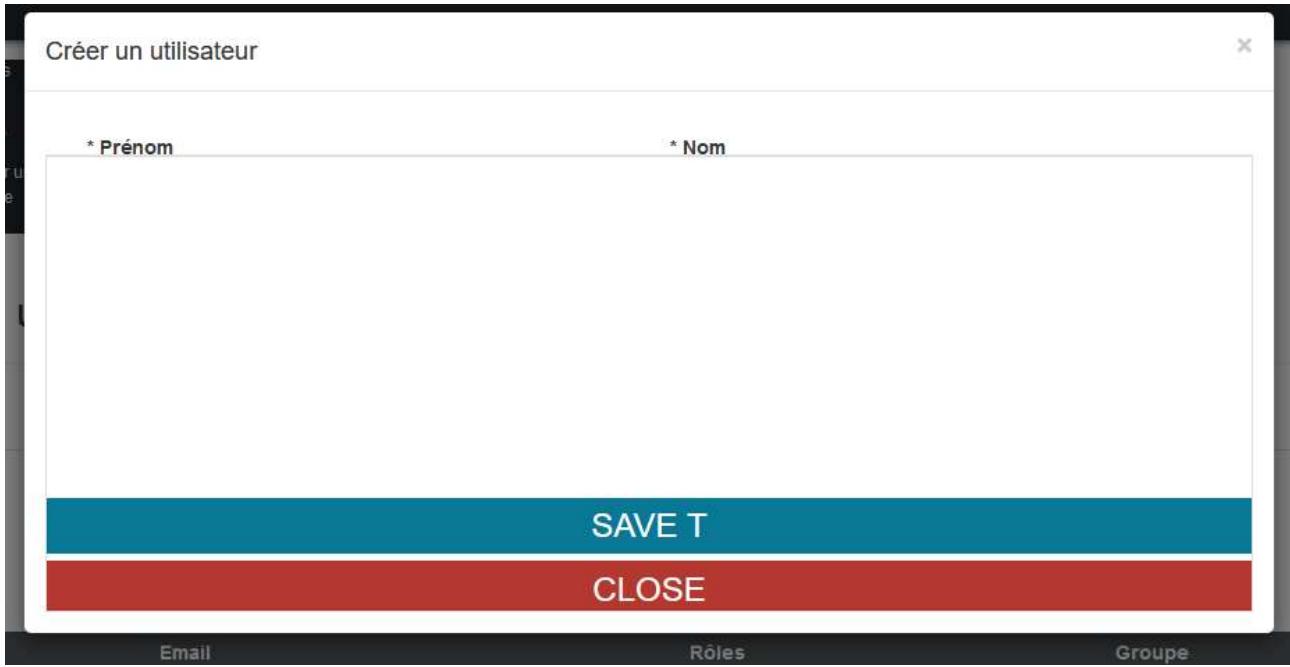
```
<div class="modal-header">
</div>
<div class="modal-body">
</div>
<div class="modal-footer btn-full-width">
  <div class="modal-sticky-btn">
    <button type="submit" class="btn btn-submit" form="product_new">
      <%= t('produits_voyages.gestion.products.new.save') %>
    </button>
    <button type="button" class="btn btn-cancel" data-dismiss="modal">
      <%= t('produits_voyages.gestion.products.new.close') %>
    </button>
  </div>
</div>
```

Le formulaire :

```
<%= simple_form_for @user, { url: url } do |f| %>
<%= render 'gestion/shared/model_errors', model: @user %> <!-- col-xs-6 -->
<%= f.input :name, autofocus: true, wrapper_html: { class: 'col-md-6' } %>
<%= f.input :lastname, wrapper_html: { class: 'col-md-6' } %>
<%= f.input :email, wrapper_html: { class: 'col-md-12' } %>
<%= f.input :password, wrapper_html: { class: 'col-xs-6' } %>
<%= f.input :password_confirmation, wrapper_html: { class: 'col-xs-6' } %>
<% if @group.blank? %>
  <% if @current_abilities.can?(:manage, ::Authentification::User) %>
    <div class="col-xs-12 col-md-6">
      <%= f.label :roles %>
      <%= f.input_field :roles, collection: @roles, as: :select, class: 'selectpicker' %>
    </div>
  <% unless @user.old_group_id.nil? %>
    <%= f.input :old_group_id, as: :hidden, wrapper: false %>
  <% end %>
  <%= f.association :group, input_html:{ class: 'selectpicker form-control', data: {}} %>
<% end %>
<% end %>
```

Avec cette modale, j'ai rencontré pas mal de problèmes.

Voici à quoi elle ressemblait initialement :



Le premier problème au niveau des inputs vient de Bootstrap, car il ne gère pas correctement les formulaires créés avec Simple Form. J'ai été obligé de séparer les inputs dans des divs avec la classe 'row'.

Ensuite, le deuxième problème concerne les boutons. Sur la capture d'écran, j'avais essayé d'afficher les valeurs avec cette commande `::Authentification::User.human_attribute_name(:save)`, mais cela n'a jamais fonctionné. Le problème venait de ma méthode. En réalité, je n'aurais pas dû créer les boutons dans la modal. Je devais utiliser un layout dédié à la modal créé par mes collègues. J'ai donc dû retirer les boutons du modal et modifier le constructeur pour lui indiquer le layout à utiliser. Voici la section "NEW" du contrôleur :

```
def new
  @modal = request.xhr?
  @modal_title = ::Authentification::User.human_attribute_name(:title)
  @modal_form = 'new_user'
  @modal_footer_full_width = true
  @modal_no_header = true
  @user = User.new
  @roles = roles_to_select(manageable_roles) if @group.blank?
  render_dependency_format :new
end
```

Vérifie si la requête est une requête ajax ; s'il ne s'agit pas d'une requête ajax, il utilisera un autre layout

```
@modal = request.xhr?
```

Définit le titre du modal

```
@modal_title = ::Authentification::User.human_attribute_name(:title)
```

Définit le formulaire qui sera exécuté pour le bouton submit

```
@modal_form = 'new_user'
```

Met à true certains paramètres. Le premier fait que les boutons dans le footer prennent toute la place en longueur

et le deuxième indique qu'il n'y a pas de header dans la modal pour que le layout utilise le sien.

```
@modal_footer_full_width = true
```

```
@modal_no_header = true
```

Instancie un utilisateur vide

```
@user = User.new
```

Définis un rôle de base s'il n'y a pas de groupe

```
@roles = roles_to_select(manageable_roles) if @group.blank?
```

Renvoie la vue avec le format adapté

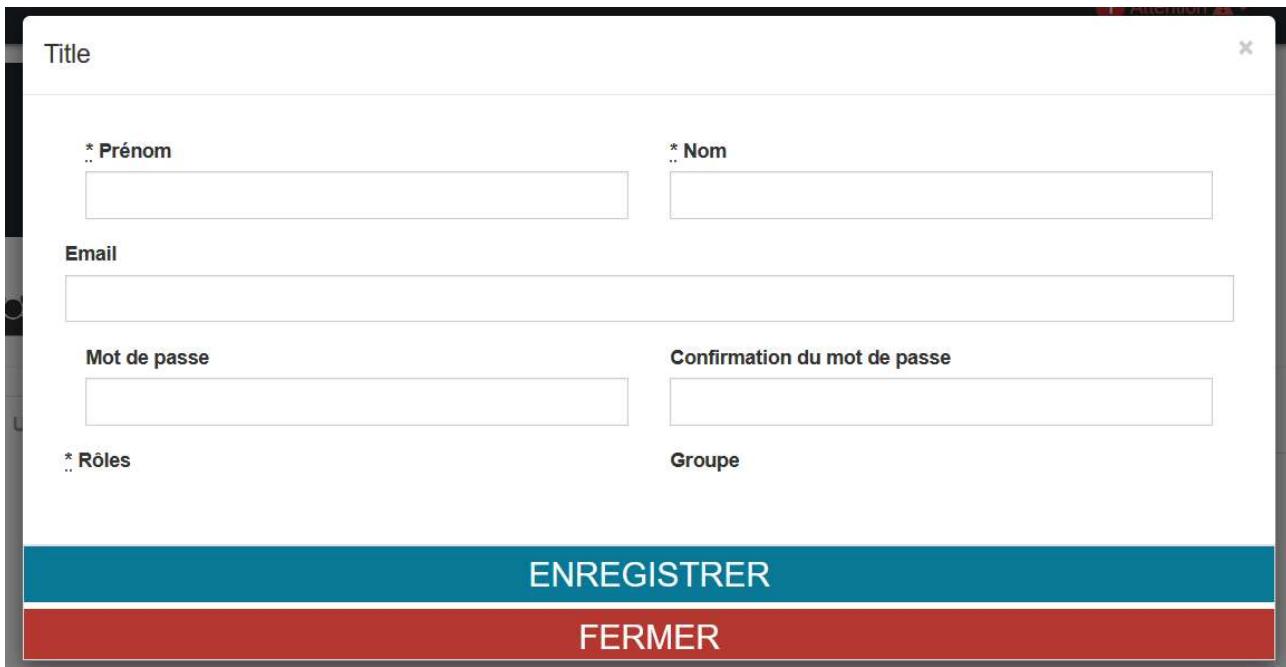
```
render_dependency_format :new
```

Il y avait également cette fonction à modifier :

```
def resolve_layout
  if [ :new, :create, :show ].include?(params[:action].to_sym)
    if request.xhr?
      "gestion/form_modal"
    else
      "authentification/gestion/users_form_page"
    end
  end
```

Il vérifie pour new, create et show si la requête est en ajax ; si c'est le cas, il renvoie le layout à utiliser, ici form_modal.

Troisième problème :

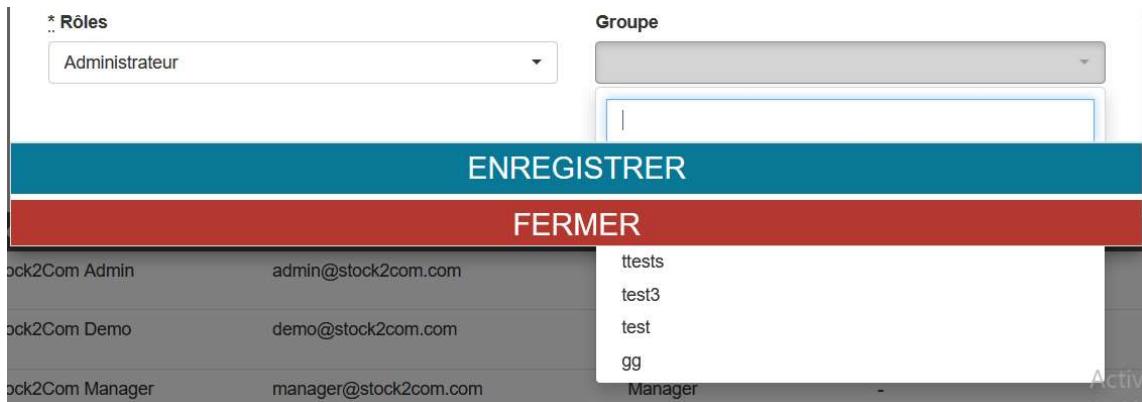


Les inputs de rôle et de groupe ne s'affichent pas correctement. Je n'ai pas réussi à résoudre ce problème moi-même, j'ai dû demander de l'aide à un collègue. Le problème vient du fait que le sélecteur utilise une bibliothèque appelée Bootstrap Select. C'est une bibliothèque JavaScript qui ajoute des fonctionnalités au sélecteur pour le rendre plus moderne. Le truc est qu'il faut initialiser la bibliothèque lors du chargement de la page, ce que je n'avais pas fait. J'ai donc dû ajouter dans la vue :

```
<script>
$(document).ready(function () {
  $('.selectpicker').selectpicker();
});
</script>
```

Dernier problème :

Mes listes déroulantes ne s'affichent pas au bon endroit. Elles apparaissent en dessous des boutons Enregistrer et Fermer.



Pour résoudre ce problème, il faut modifier l'attribut CSS z-index et lui donner une valeur plus grande que celle des boutons.

Au début, j'ai essayé de le mettre directement dans le input :

```
<%= f.input_field :roles, collection: @roles, as: :select, class: 'selectpicker form-control col-sm-6',  
style: 'z-index: 9999;', data: { live_search: true }, title: "", include_blank: false %>
```

Cependant, cela ne fonctionne pas car ce n'est pas compatible avec l'attribut selectpicker. Cette ligne est le conteneur qui va générer le code HTML. Mon CSS nécessite que je l'applique au HTML généré par selectpicker.

Le code CSS final :

```
.modal-body .dropdown-menu {  
z-index: 3000;  
}
```

Ainsi, je sélectionne uniquement les éléments dropdown-menu qui se trouvent dans un modal-body afin de ne pas affecter les autres listes déroulantes qui pourraient ne pas en avoir besoin.

Après avoir terminé le formulaire, je suis passé à la section dédiée au groupe :



Aymeric Moreau

Dans cette section, je dois remplacer la barre de recherche par un accordéon qui affichera tous les groupes. Ainsi, je vais chercher un modèle d'accordéon sur une autre page. Sur la page de gestion des produits, j'ai trouvé un accordéon qui correspond à mes besoins.

Voici la structure de l'accordéon :

```
<p>
  <div class="panel panel-gray no-vertical-margin">
    <div class="panel-heading pointer collapsed strong" data-toggle="collapse" data-target="#searchContent" aria-expanded="false">
      <strong>
        <%= ::Authentification::Group.human_attribute_name(:title) %>
      </strong>
      <%= icon("chevron-down fa-2x pull-right") %>
    </div>
    <div id="searchContent" class="panel-collapse collapse" aria-expanded="false" style="width: 100%">
      <div class="panel-body">

      </div>
    </div>
  </div>
</p>
```

Pour l'intérieur de l'accordéon, je vais réutiliser le tableau que j'utilise pour mes utilisateurs en l'adaptant pour les groupes.

Le tableau est le suivant :

```
<table class="table table-header-gray table-condensed"> <!-- table-bordered table-striped -->
  <tbody>
    <% @groups.each do |group| %>
      <tr>
        <td><%= group.name %></td>
        <td class="text-right">
          <!-- gestion_index_dynamic_abilities_values_path -->
          <%= link_to gestion_index_dynamic_abilities_values_path(group), class: "btn btn-restrict", data: {"toggle" => "tooltip", "placement" => "top"}, title: ::Authentification::Group.human_attribute_name(:edit) do %>
            <%= ::Authentification::Group.human_attribute_name(:restriction) %>
          <% end %>

          <%= link_to edit_gestion_group_path(group), class: "btn btn-edit", data: {"toggle" => "tooltip", "placement" => "top"}, title: ::Authentification::Group.human_attribute_name(:edit) do %>
            <%= t('global.edit') %>
          <% end %>

          <%= link_to gestion_group_path(group), class: "btn btn-show", target: "_blank", data: {"toggle" => "tooltip", "placement" => "top"}, title: ::Authentification::Group.human_attribute_name(:show) do %>
            <%= t('global.show') %>
          <% end %>
          <!-- ::Authentification::Group.human_attribute_name(:show) -->
          <%= link_to gestion_group_path(group), method: "delete", class: "btn btn-destroy delete-modal", data: {"toggle" => "tooltip", "placement" => "top", "modal-remote-links" => true, "delete-message" => sprint( ::Authentification::Group.human_attribute_name(:confirm_delete))}, title: ::Authentification::Group.human_attribute_name(:destroy) do %>
            <%= ::Authentification::Group.human_attribute_name(:destroy) %>
          <% end %>
        </td>
      </tr>
    <% end %>
  </tbody>
</table>
```

Dans le tableau, nous parcourons tous les groupes et affichons leur nom. Ensuite, pour chaque groupe, nous ajoutons des boutons de restriction, d'édition, d'en savoir plus et de suppression.

Voici à quoi cela ressemble sur le site :

Title	Restriction	Modifier	Voir	Supprimer
gg				
test				
test				

Pour finir la semaine, je me suis occupé du formulaire de création de groupe. Il n'était pas dans le modèle, mais j'ai ajouté un bouton similaire à "créer un utilisateur".

```
<div class="row">
  <div class="col-xs-12 col-sm-12 col-md-2 col-md-offset-10">
    <div class="panel panel-green panel-no-border">
      <div class="panel-heading">
        <div class="row">
          <div class="col-xs-12 text-center">
            <%= link_to new_gestion_group_path , class: 'ajax-modal text-decoration-none' do %>
              <%= icon "plus fa-3x" %>
              <h3 class="no-vertical-margin">
                | <%= t("authentication.gestion.groups.index.add") %>
              </h3>
            <% end %>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
```

Ensuite, j'ai modifié le contrôleur pour que le formulaire s'affiche dans une modale :

```
def new
  @group = Group.new
  @modal = request.xhr?
  @modal_title = ::Authentication::Group.human_attribute_name(:title)
  @modal_form = 'group_form'
  @modal_footer_full_width = true
  @modal_no_header = true
end
```

```
def resolve_layout
  if [ :new, :create, :show ].include?(params[:action].to_sym)
    if request.xhr?
      "gestion/form_modal"
    else
```

Aymeric Moreau

Je n'ai rien à modifier dans le fichier de formulaire, donc le formulaire est terminé :

Title

* Nom

ENREGISTRER

FERMER

Après cette semaine de travail bien fourni, il me reste deux tâches importantes à accomplir sur ce projet : la modale de restriction et le regroupement des deux sections en une.

Je pense pouvoir les terminer la semaine prochaine.

Voici le tableau Trello à la fin de cette première semaine :



Dans la carte groupe la seule tâche non faite c'est la modal restriction

Semaine 3

Cette semaine, j'avais prévu de terminer ma page, mais j'ai été trop optimiste.

J'ai commencé par m'occuper de la page dédiée à la gestion des restrictions pour un groupe. Cependant, je me suis rendu compte que la page était déjà réalisée, mais je ne l'avais tout simplement pas trouvée sur le site de base. En fait, tout ce que j'avais à faire, c'était de la déplacer dans une modale.

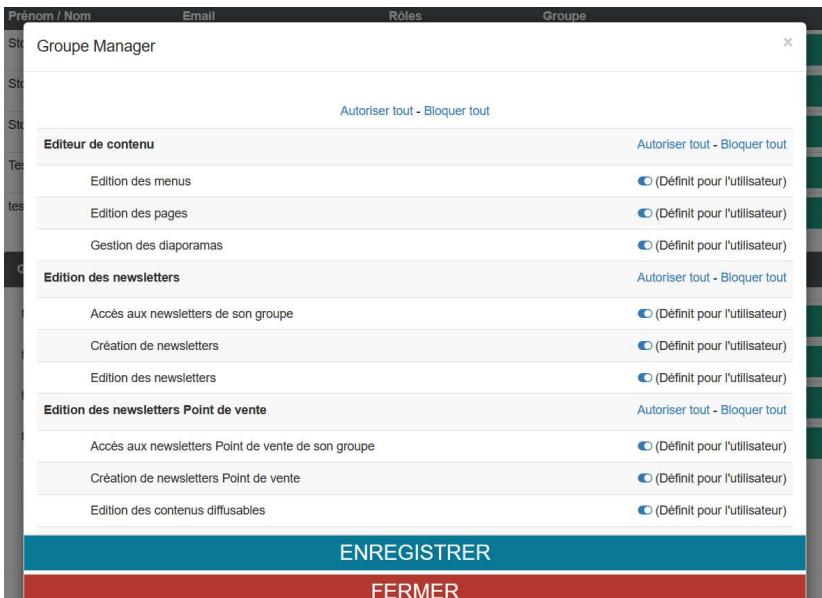
Ainsi, dans le contrôleur Dynamic_Abilities_Values_Controller, qui gère les restrictions, j'ai dû modifier la section index en ajoutant les attributs pour la modale et en changeant le type de requête :

```
def index
  @dynamic_abilities = @target.list_abilities
  @modal = request.xhr?
  @modal_title = t('authentication.gestion.dynamic_abilites_values.index.title')
  @modal_form = ""
  @modal_footer_full_width = true
  @modal_no_header = true
  renderDependingFormat :index
end
```

J'ai également dû modifier la fonction resolve_layout pour définir le layout à utiliser :

```
def resolve_layout
  if request.xhr?
    "gestion/form_modal"
  else
    ...
  end
```

Voici comment la modale s'affiche :



Ce n'était pas visible dans la modale ni dans le code, mais il y avait un problème avec les modales. Les titres que j'avais configurés dans les constructeurs ne s'affichaient pas. Le souci résidait dans l'utilisation de la ligne que j'avais mentionnée lors de la première semaine : ::Authentication::User.human_attribute_name(:title). Cette ligne ne fonctionne pas dans le contrôleur. Je pense que la raison en est que la fonction human_attribute_name n'est pas configurée pour les contrôleurs, mais plutôt pour les vues. Par conséquent, je suis contraint d'utiliser t().

Après avoir finir le modal pour les restrictions j'ai commencé a regardé pour rassembler tous les éléments sur la même page.

J'ai commencé par de la review de code pour comprendre comment l'affichage des parties utilisateur et groupe fonctionne.

Le code qui gère le menu de navigation entre les deux sections est le suivant :

```
<% content_for :css do %>
| <%= stylesheet_link_tag 'authentification/gestion/application', media: 'all' %>
<% end %>

+ <% content_for :js do %>
| <%= javascript_include_tag 'authentification/gestion/application' %>
<% end %>
-<% content_for :content do %>
- <%= page_title t('authentification.gestion.users.tabs.title'), 'users' %>

- <div role="tabpanel">

-   <ul class="nav nav-tabs" role="tablist">
-     <%= render 'gestion/shared/tabli', text_index: 'authentification.gestion.users.tabs', id: 'users', active: (@tab_active == 'users' || @tab_active.nil?), url: gestion_users_path() %>
-     <% if @current_abilities.can? :manager, ::Authentication::Group %>
-       <%= render 'gestion/shared/tabli', text_index: 'authentification.gestion.users.tabs', id: 'groups', active: (@tab_active == 'groups'), url: gestion_groups_path() %>
-     <% end %>
-   </ul>

-   <div class="tab-content">
-     <%= render 'gestion/shared/tabpanel', id: 'users', active: (@tab_active == 'users' || @tab_active.nil?) %>
-     <% if @current_abilities.can? :manager, ::Authentication::Group %>
-       <%= render 'gestion/shared/tabpanel', id: 'groups', active: (@tab_active == 'groups') %>
-     <% end %>
-   | </div>
- </div>

<% end %>

<%= render 'gestion/shared/application_layout' %>
```

Il est donc nécessaire de supprimer ce code. Pour le moment, j'ai conservé le fichier et remplacé le menu par :

```
<div class="user-section">
  <%= render 'gestion/shared/tabpanel', id: 'users', active: true %>
</div>
```

Cela affiche maintenant uniquement la page utilisateur sans avoir de bouton pour changer de section.

Ensuite, j'ai regroupé tout l'affichage dans l'index d'utilisateur. Ainsi, dans le contrôleur utilisateur, j'ai ajouté le code qui se trouvait dans l'index du contrôleur de groupe.

Aymeric Moreau

```
def index
  @groupss = Group.order(:name).paginate(page: params[:page])
  @change_page = {}

  if @groupss.blank?
    @change_page = {'data-s2c-remote-links-from': params[:page] }
    @change_page['data-s2c-remote-links-to'] = params[:page] = [ (@groupss.total_entries.to_f / @groupss.per_page).ceil, 1 ].max
    @groupss = @groupss.paginate(page: params[:page])
  end

  @search_params = search_params
  users = User.accessible_by(@current_abilities, :read).order(:lastname, :name)
  users = users.search_name_or_email(@search_params[:name_like]) unless @search_params[:name_like].blank?
  if @group.blank?
    users = users.search_roles(@search_params[:roles]) unless @search_params[:roles].blank?
    users = users.search_groups(@search_params[:groups]) unless @search_params[:groups].blank?
  else
    users = users.search_groups([@group.id])
  end
  @users = paginate_list users

  if @group.blank?
    @roles = roles_to_select( readable_roles )
    @groups = Group.all
  end

  renderDependingFormat :index
end
```

Dans le fichier _index de l'utilisateur, j'ai rendu l'affichage des groupes avec :

```
<%= render partial: 'authentification/gestion/groups/groupsCollapse' %>
```

De plus, j'ai ajouté le bouton "Créer un groupe" dans le même fichier que le bouton de création d'un utilisateur.

Voici à quoi ressemble la page maintenant :

Aymeric Moreau

The screenshot shows the 'Gestion des utilisateurs' (User Management) page in the Stock2Com administration interface. The left sidebar contains various navigation links such as Accueil, Devis / Réservations, Messages / Avis, Produits, Site > Configuration, Utilisateurs, Site > Apparence, Site > Contenus, Site > Blog, Site > Outils, Newsletters, Boîtes mail, Journal des événements, and contact information (stock2com@stock2com.com, 06 63 27 57 05). The main content area has three buttons at the top: 'Créer une destination' (Product), 'Rédiger un article' (Blog), and 'Créer une sélection de produits' (Links). Below these are two large green buttons: '+ Crée un utilisateur' and '+ Crée un groupe'. The main table lists users with columns for Nom / Prénom, Email, Rôles, Groupe, and actions (Modifier, Voir, Supprimer). A secondary table below shows 'Gestion des Groupes / Restriction d'accès' with rows for various user names and their associated group restrictions.

Nom / Prénom	Email	Rôles	Groupe			
azerty azerty	azerty@gmail.com	Administrateur	-			
brtrb fberfb	erfg@gmail.com	Administrateur	-			
9999 99999	99999@gmail.com	Administrateur	-			
Stock2Com Admin	admin@stock2com.com	Administrateur	-			
Stock2Com Demo	demo@stock2com.com	Administrateur	-			
Stock2Com ManagerB	manager@stock2com.com	Manager	-			
Stock2Com User	user@stock2com.com	Utilisateur Gestion	-			
tegd Téle	telf@gmail.com	Administrateur	-			
Test Test211	testtest@gmail.com	Utilisateur Gestion	test3			
test testd	test@gmail.com	Administrateur	-			

Gestion des Groupes / Restriction d'accès			
753			
abestest			
abestest			
baghv			
dhf			
eghreh			
egs			
ernjhs			
tgf			
ggfk			

← Précédent 1 2 3 4 Suyivant →

Avec la fusion des deux parties de la page, j'ai dû apporter des modifications au niveau du formulaire. Lorsque l'on clique sur "Voir un groupe", on arrive sur une page avec le nom du groupe, le nombre d'utilisateurs associés et un bouton pour revenir à la liste des groupes :

[Retour aux groupes](#)

Fiche du groupe ggfk

Nom ggfk

Utilisateurs associés 0

J'ai dû modifier le lien du bouton de retour en arrière dans le layout :

```
<% end %>
<% content_for :content do %>
  <%= back_button(gestion_users_path, t('authentication.gestion.groups.form.back')) %>
  <%= page_title t("authentication.gestion.groups.#{params[:action]}.title", name: ( !@group.nil? ? @group.name : nil ), 'users' %>
  <%= yield %>
<% end %>
<%= render 'gestion/shared/application_layout' %>
```

Mais cela s'est révélé inutile en raison d'un changement dans le projet. Désormais, le bouton "Voir" doit ouvrir une modale.

Dans le constructeur, à la section "show", j'ai donc dû ajouter les attributs de la modal :

```
def show
  @modal = request.xhr?
  @modal_title = t('authentication.gestion.groups.show.title', name: @group.name)
  @modal_no_header = true
end
```

Le layout est déjà bien configuré dans le résolve layout. Il me reste plus simplement qu'à ajouter "ajax-modal" au bouton "show".

```
<%= link_to gestion_group_path(group), class: "ajax-modal btn btn-show", target: "_blank", data: {"toggle" => "tooltip", "placement" => "top"}, title: t("authentication.gestion.groups.index.show") do %> <!-- ::Authentification::Group.human_attribute_name(:show) -->
```

J'ai dû faire la même chose pour le bouton voir d'user.

Voici le tableau Trello à la fin de cette semaine :



Il y a eu des tâches de rajouter :

Déjà le déplacement de la page show dans une modal que j'ai déjà coché mais il y a eu d'autre nouveauté comme la mise en place d'une actualisation ciblé lors de la modification ou l'ajout d'un utilisateur/group et la refont du modale de confirmation de suppression.

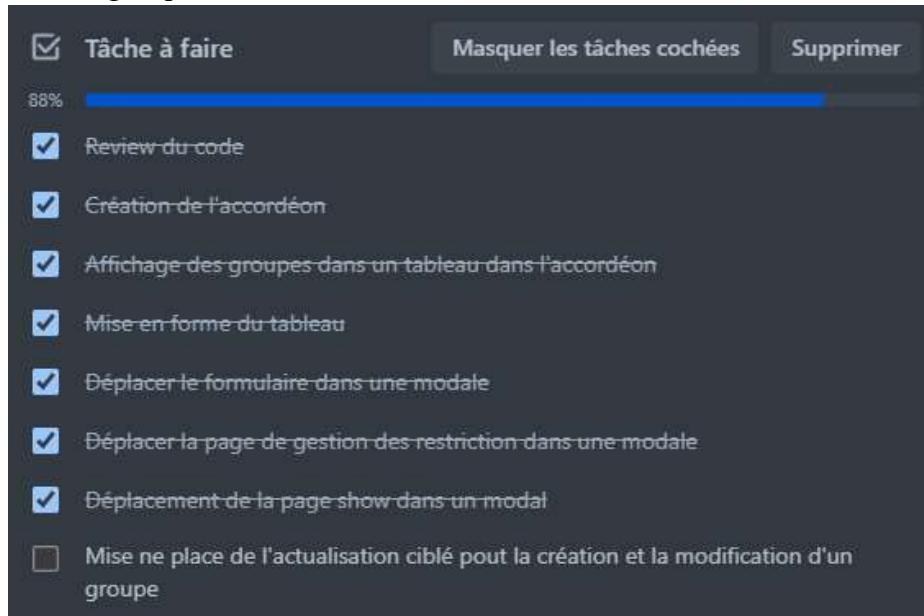
Voici la carte utilisateur :

This is a screenshot of a Trello task list for the "Utilisateur" card in the "Terminer" column. The list includes the following tasks:

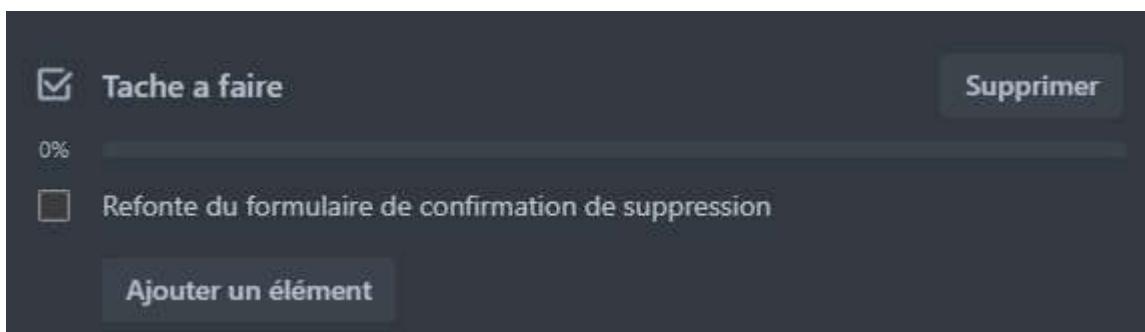
- Tâche à faire
- Review du code
- Affichage des utilisateurs dans un tableau
- Mise en forme du tableau
- Déplacement du formulaire dans une modal
- Déplacement de la page show dans une modal
- Mise en place de l'actualisation ciblé pour la création et la modification d'un User

At the top, there are buttons for "Masquer les tâches cochées" and "Supprimer". A progress bar shows 83% completion. At the bottom, there is a button for "Ajouter un élément".

la carte groupe :



Et la carte modification global :



Semaine 4

Cette semaine, j'ai débuté en m'occupant de l'optimisation de l'affichage du site afin de réduire la consommation de ressources. L'optimisation consiste à actualiser uniquement les parties pertinentes de la page lors de l'ajout ou de la modification d'un élément, comme un utilisateur ou un groupe, plutôt que de rafraîchir la page entière. Pour illustrer, voici comment j'ai procédé avec le tableau des utilisateurs :

Il y a deux fonctionnalités à implémenter : lors de la création d'un utilisateur, seul le tableau est actualisé, et en cas de modification d'une valeur dans le tableau, seule la ligne modifiée doit être mise à jour.

La première étape consiste à retirer le tableau du fichier index et à le diviser en deux fichiers distincts. Le premier sera dédié au tableau en lui-même, tandis que le deuxième sera consacré à une ligne spécifique du tableau. Il sera donc nécessaire d'utiliser une boucle foreach pour rendre la partial autant de fois qu'il y a d'utilisateurs.

Voici comment cela se traduit pour le fichier _user_table :

```
<div>
<% unless @users.blank? %>
  <%= will_paginate @users, {renderer: BootstrapPagination::Rails, link_options: {'data-s2c-remote-links': true}}.merge(@change_page) %>

  <table id="tableUser" class="table table-header-gray table-condensed" > <!-- table-bordered table-striped -->
    <thead>
      <tr>
        <th>
          | <%= ::Authentification::User.human_attribute_name(:lastname) %> / <%= ::Authentification::User.human_attribute_name(:name) %>
        </th>
        <th class="">
          | <%= ::Authentification::User.human_attribute_name(:email) %>
        </th>
        <th class="">
          | <%= ::Authentification::User.human_attribute_name(:roles) %>
        </th>
        |   <th class="text-center">
          |     <%= ::Authentification::User.human_attribute_name(:group) %>
        </th>

        <th>
        </th>
      </tr>
    </thead>
    <tbody>
      <% @users.each do |user| %>
        <tr id="user-<%= user.id %>">
          |   <%= render partial: 'authentification/gestion/users/user_table_row', locals: {user: user} %>
        </tr>
      <% end %>
    </tbody>
  </table>

  <%= will_paginate @users, renderer: BootstrapPagination::Rails, link_options: {'data-s2c-remote-links': true} %>
<% else %>
  <p class="alert alert-warning col-xs-8 col-xs-offset-2">
    <%= t('gestion.commun.aucun_resultats') %>
    <% unless @search_params.empty? %>
      <%= link_to @group.blank? ? gestion_users_path : gestion_group_users_path(@group), class: 'btn btn-default pull-right' do %>
        |   <i class="fa fa-history"></i> <%= t('gestion.commun.reinitialiser_filtres') %></i>
      <% end %>
    <% end %>
  </p>
<% end %>
</div>
```

Les seules choses qui changent sont que l'affichage du contenu du tableau se fait par un rendu (render), et que pour pouvoir sélectionner un utilisateur spécifique dans le code JavaScript, j'ai ajouté un identifiant (id) à la balise "tr" correspondant à l'ID de l'utilisateur.

Et mon user_table_row pour le contenu du tableau :

Aymeric Moreau

```
<td>
    <%= user.lastname %>
    <%= user.name %>
</td>
<td><%= user.email %></td>
<td>
    <% user.role_symbols.each_with_index do |role,index| %>
        <%= t "activerecord.attributes.authentication/user.roles_symbols.#{role}" %>
        <%= ' / ' unless index == user.role_symbols.size - 1 %>
    <% end %>
</td>
<td><%= !user.group.nil? ? user.group.name : '-' %></td>
<td class="text-right">

    <%= link_to edit_gestion_user_path(user), class: " ajax-modal btn btn-edit", data: {"toggle" => "tooltip", "placement" => "top"}, title: ::Authentication::User.human_attribute_name(:edit) do %>
        <div class="btn-label"> <%= t('global.edit') %> </div> <!-- ajax-modal -->
    <% end %>

    <%= link_to gestion_user_path(user), class: " ajax-modal btn btn-show", target: "_blank", data: {"toggle" => "tooltip", "placement" => "top"}, title: ::Authentication::User.human_attribute_name(:show) do %>
        <div class="btn-label"> <%= t('global.show') %> </div>
    <% end %>
        <!-- delete-modal -->
        <%= link_to gestion_user_path(user), method: "delete", class: "btn btn-destroy ajax-modal delete-modal", data: {"toggle" => "tooltip", "placement" => "top", "modal-remote-links" => true, "delete-message" => sprintf( ::Authentication::User.human_attribute_name(:confirm_delete))), title: ::Authentication::User.human_attribute_name(:destroy) do %>
    <% end %>

</td>
```

Après de cette occupé des vue il faut s'occupé du javascript car c'est lui qui vas permettre l'actualisation ciblé.

Il faut créer 2 fichiers Create.js.erb et Uptdate.js.erb c'est sur ces fichiers que l'on va rediriger dans le contrôleur une fois la création ou la modification faite.

Voici le fichier Create.js.erb :

```
<% if @user.valid? %>
    $('#ajaxModal').modal('hide');

    $('#tableUser').html("<%= j(render(partial: 'authentication/users/user_table')) %>");

    $('#tableUser').ready(function(event){
        S2C_Gestion.Base.initPage('#tableUser');
    });

<% end %>
```

Donc on commence par vérifier si l'utilisateur créer est valide ensuite on ferme la modal, on régénère le tableau sur l'élément qui a comme id tableUser (lui-même) et on désactive la boucle de chargement.

Et voici le fichier Update.js.erb :

```
<% if @user.valid? %>
  $('#ajaxModal').modal('hide');
  $('#user-<%= @user.id %>').html("<%= j(render(partial: 'authentification/gestion/users/user_table_row', locals: {user: @user})) %>");
  $('#user-<%= @user.id %>').ready(function(event){
    S2C_Gestion.Base.initPage('#user-<%= @user.id %>');
  });
<% end %>
```

Donc il ferme la modale, il régénère la ligne du tableau modifier et il désactive la boucle de chargement.

Au début je pensais que faire cela était suffisant, mais il y a eu un problème impossible de faire exécuter le script js à la création au à la modification de l'utilisateur.

Le problème vient de mon formulaire, il est incomplet et ne se lance pas en ajax. Pour résoudre mon problème, j'ai dû modifier l'ouverture du formulaire :

```
<%= globalize_simple_form_for @user, { url: url, remote: true, html: { id: 'user_form' }, data: { turbo: false } } do |f| %>
```

L'URL est définie avec ce code :

```
<%
  if @user.new_record?
    url = gestion_users_path
  else
    url = gestion_user_path(@user.id)
  end
%>
```

Ensuite, remote: true active la soumission ajax du formulaire. Turbo: false désactive une fonctionnalité de Hotwire Turbo qui posait problème dans ce formulaire.

Une fois cela sauvegardé, l'actualisation ciblée du formulaire fonctionne. Ensuite, je me suis chargé d'améliorer la modale de suppression :



Il faut que je modifie les boutons pour qu'ils soient en accord avec les nouvelles modales. Pour ce faire, j'ai dû trouver l'endroit où se trouvait cette modale, car le bouton supprimer n'est pas fait de la

même manière que les autres ; il n'y a pas de fichier "destroy" dans les vues. L'affichage de la modal se fait uniquement grâce à l'attribut "delete-modal".

```
<%= link_to gestion_group_path(group), method: "delete", class: "btn btn-destroy delete-modal",
| data: {"toggle" => "tooltip", "placement" => "top", "modal-remote-links" => true,
| "delete-message" => sprintf( ::Authentification::Group.human_attribute_name(:confirm_delete)), 
| title: t("authentification.gestion.groups.index.destroy") do %>
<% end %>
```

Cet attribut est lié au fichier "modal-delete.coffee" (le CoffeeScript est un langage qui se compile en JavaScript) ; c'est ce fichier qui va gérer la modal. Il appelle une vue appelée "delete_modal.html.erb" :

```
<div id="deleteModal" class="modal fade">
<div class="modal-dialog">
<div class="modal-content">
<div class="modal-header">
<button type="button" class="close" data-dismiss="modal" aria-label="<%= t('gestion.gestion.modal.close') %>">
<span aria-hidden="true">&times;</span></button>
<h4 class="modal-title" data-default="<%= t('gestion.gestion.modal.delete.title') %>"></h4>
</div>
<div class="modal-body" data-default="<%= t('gestion.gestion.modal.delete.body') %>"></div>
<div class="modal-footer ">
<%= link_to t('gestion.gestion.modal.delete.footer.delete'), '', method: :delete, class: "btn btn-destroy modal-valid hidden",
| data: {dismiss: 'modal' } %>
<%= button_to t('gestion.gestion.modal.delete.footer.delete'), '', method: :delete, class: "btn btn-destroy modal-valid-remote hidden",
| form: { data: { s2c_formhelper: 'remote' } } %>
| <button type="button" class="btn btn-suspend" data-dismiss="modal"><%= t('gestion.gestion.modal.delete.footer.cancel') %></button>
</div>
</div>
</div>
</div>
```

Donc, c'est une modal classique avec un titre, un bouton de fermeture, un bouton "submit" et un bouton "annuler". J'ai deux changements à faire : d'abord, je dois actualiser le style des boutons pour qu'ils correspondent au nouveau standard de couleur et de taille de texte, et je dois étendre les boutons pour qu'ils prennent toute la largeur.

Pour le style du bouton "submit", j'ai dû changer le tag "btn-danger" en "btn-destroy", et pour le bouton "annuler", il n'y avait pas de classe scss définie, donc j'ai dû en créer une appelée "btn-suspend".

```
.btn.btn-suspend{
  color: $unclassified;
  background-color: $default;
  border-radius: 0;
  font-size: 18px;
  height: 36px;
}
```

Je l'ai appelée ainsi parce qu'il y avait déjà un bouton "cancel" qui servait pour d'autres boutons mais qui ne convenait pas pour celui-là.

Il change donc la couleur du texte pour la couleur qui est liée à cette variable scss, qui est le noir. Il change la valeur du background pour du blanc, qui est la couleur liée à cette variable que j'ai moi-même ajoutée, vu qu'il n'y en avait pas.

Il enlève les bordures, définit une taille en pixels pour le texte et change la taille du bouton. Ensuite, pour que les boutons prennent toute la largeur de la modal, j'ai ajouté à la div, en prenant exemple sur la modal déjà existante : "modal-sticky-btn btn-full-width".

Voici le résultat :



À noter que le design du bouton "annuler" n'est pas définitif.

Ensuite en testant le bouton de suppression, j'ai remarqué que la modal de confirmation ne fonctionnait en fait pas ; elle supprime l'objet même si j'appuie sur "annuler".

Pour résoudre ce problème, j'ai d'abord cherché s'il y avait un bouton similaire au mien qui fonctionnait encore.

J'en ai trouvé un dans la page de gestion des coordonnées. Voici le code des deux boutons :

```
<%= link_to gestion_user_path(user), class: 'btn btn-destroy delete-modal', data: {modal_remote_links: true, method: "DELETE", confirm: t('gdonnees._pv.confirmation')} do %>
<% end %>

<%= link_to gestion_user_path(user), method: "delete", class: "btn btn-destroy delete-modal",
  data: {"toggle" => "tooltip", "placement" => "top", "modal-remote-links" => true,
  "delete-message" => sprintf( ::Authentification::User.human_attribute_name(:confirm_delete)), title: ::Authentification::User.human_attribute_name(:destroy) do %>
```

Le bouton du haut est le nouveau qui fonctionne correctement, et celui en dessous est l'ancien. Il n'y a pas beaucoup de différence entre les deux, si ce n'est la manière dont les données sont attribuées, qui change.

Après avoir terminé cela, j'ai reçu deux nouvelles tâches :

S'occuper de l'actualisation ciblée pour la modale de restriction. Actuellement, quand je clique sur un bouton, cela actualise entièrement la page. Il faut que seul le bouton sélectionné soit actualisé.

Ensuite, il y a un problème à régler avec l'affichage des erreurs lorsqu'on ajoute un utilisateur ou un group invalide. Il ne s'affiche pas correctement. Il me renvoie sur une page avec le formulaire alors qu'il devrait rester sur la modal.

Voici le tableau trello à la fin de cette semaine :



Tache rajouter pour la carte utilisateur :

- Mise en place de l'actualisation ciblé pour la création et la modification d'un User
- Régler le problème d'affichage des erreurs

Ajouter un élément

Tache rajouter pour la carte group :

- Mise en place de l'actualisation ciblé pour la création et la modification d'un groupe
- Régler le problème d'affichage des erreurs
- Mise en place de l'actualisation ciblé pour la page restriction

Ajouter un élément

Semaine 5

Cette semaine j'aimerais enfin finir ce projet.

J'ai commencé par m'occuper de l'actualisation ciblée de la modale restriction.

Voici le code de base :

```
<% unless @dynamic_abilities.blank? %>
  last_group_id = 0
<%>
<div id="abilitys-list" class="table-responsive">
  <table class="table table-striped">
    <tbody>
      <% @dynamic_abilities.each do |ability| %>
        <% if last_group_id != ability.dynamic_abilities_group_id %>
          <% unless ability.dynamic_abilities_group.blank? %>
            <tr>
              <td><label><%= ability.dynamic_abilities_group.name %></label></td>
              <td class="text-right">
                <%= link_to t('authentification.gestion.dynamic_abilites_values.index.allow_all'), 
                gestion_update_all_dynamic_abilities_values_path(@target.id,@target_type, can: 1, dynamic_abilities_group_id: ability.dynamic_abilities_group_id),
                data:{ s2c_remote_links: 'avoid_change_url' } %> -
                <%= link_to t('authentification.gestion.dynamic_abilites_values.index.disallow_all'),
                gestion_update_all_dynamic_abilities_values_path(@target.id,@target_type, can: 0, dynamic_abilities_group_id: ability.dynamic_abilities_group_id),
                data:{ s2c_remote_links: 'avoid_change_url' } %>
              </td>
            </tr>
          <% end %>
          <% last_group_id = ability.dynamic_abilities_group_id %>
        <% end %>
        <tr>
          <td><div class="col-offset-1 col-xs-offset-1"><%= ability.name %></div></td>
          <td class="text-right" id="abilities-<%= ability.dynamic_abilities_group_id %>-<%= ability.id %>">
            <%= link_to gestion_update_dynamic_abilities_value_path(@target.id,@target_type, ability.id, ability.can.to_i == 1 ? 0 : 1),
            data:{ s2c_remote_links: 'avoid_change_url' } do %>
              <% icon ability.can.to_i == 1 ? 'toggle-on' : 'toggle-off' %>
            <% end %>
            (<%= t "authentification.gestion.dynamic_abilites_values.index.inherits.#{ability.can_origin}" %>)
          </td>
        </tr>
      <% end %>
    </tbody>
  </table>
</div>
<% else %>
  <div class='alert alert-warning'><%= t 'authentification.gestion.dynamic_abilites_values.index.empty_abilities' %></div>
<% end %>
```

Pour cela j'ai tout d'abord séparer le formulaire du fichier _index.

J'ai donc créé un fichier appelé _dynamic_abilities_form pour accueillir le formulaire. Le but de ce fichier est qu'à chaque fois qu'un bouton est actionné, il actualise seulement le formulaire. Pour cela, j'ai dû ensuite créer un fichier update.js.erb avec ce code :

```
// Actualiser le formulaire restriction
$('#da_form').html("<%= j(render(partial: 'authentification/gestion/dynamic_abilities_values/dynamic_abilities_form',
| locals: {dynamic_abilities: @dynamic_abilities, target: @target , target_type: @target_type} )) %>");

// désactive la roue de chargement
$('#da_form').ready(function(event){
  S2C_Gestion.Base.initPage('#da_form');
});
```

Ensuite j'ai ajouté remote : true à chaque input du formulaire pour que la requête qu'il envoie soit en ajax.

Et pour finir, dans le contrôleur, j'ai redirigé les sections update et update all vers le fichier update avec un render depending format :update.

Après avoir terminé cela, je me suis occupé de l'affichage des erreurs dans les modales. Cette tâche fut assez simple, il suffit d'ajouter un else à la condition if @group.valid? dans les fichiers JavaScript, avec à l'intérieur un code pour actualiser le formulaire avec la variable du groupe en paramètre :

L'affichage des erreurs se fait grâce à cette ligne : <%= render 'gestion/shared/model_errors', model: @group %>. Sa fonction est d'afficher les messages d'erreurs qui se trouvent dans la variable @group.

```
<% else %>
  $('#group_form').html("<%= j(render(partial: 'authentification/gestion/groups/form', locals: { group: @group })) %>");
  S2C_Gestion.Base.initPage('#group_form');
<% end %>
```

Ensuite, dans le contrôleur, j'ai dû simplifier les actions create et update car dans le code de base, il y a deux render : un pour les erreurs et un autre pour quand cela fonctionne correctement. J'ai simplifié en ne laissant qu'un seul, vu que la gestion de la réussite et de l'erreur sont dans le même fichier.

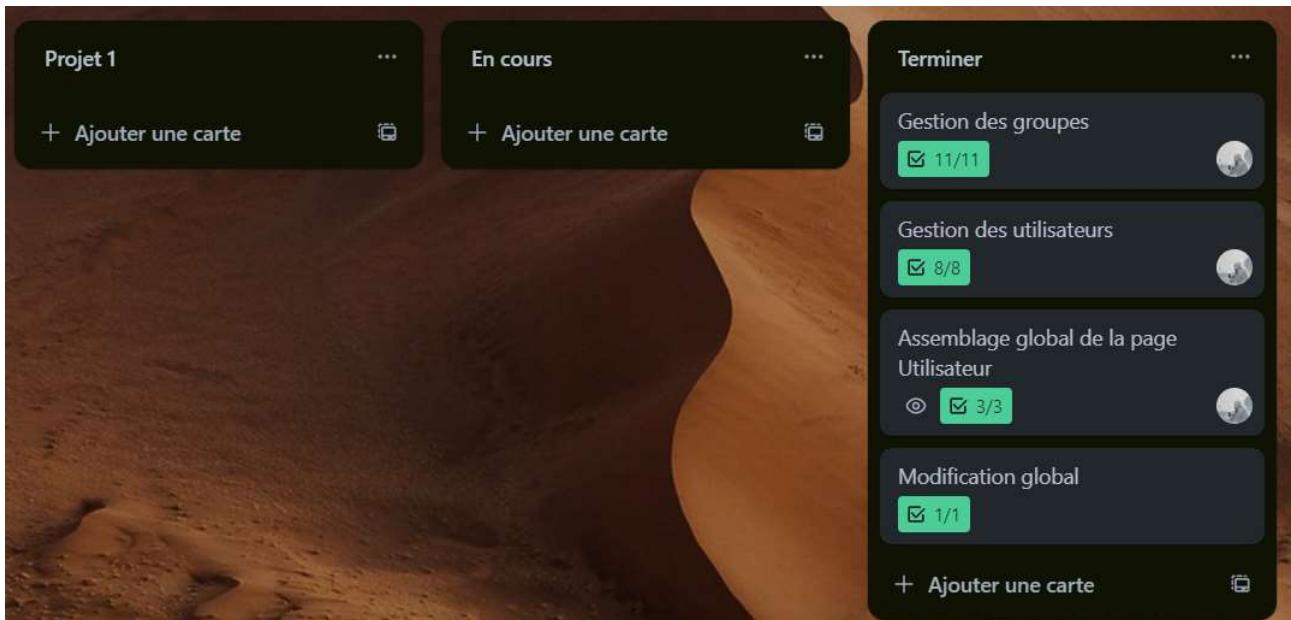
Juste après, je me suis occupé d'une nouvelle tâche à faire qui s'est encore ajoutée. Je dois configurer l'actualisation ciblée pour la suppression d'un utilisateur ou d'un groupe. Pour cela, j'ai dû, comme pour les autres actualisations, créer un fichier JavaScript appelé destroy.js. Voici le code du fichier :

```
$('#ajaxModal').modal('hide');
$('#accordionGroups').html("<%= j(render(partial: 'authentification/gestion/groups/groupsCollapse')) %>");
S2C_Gestion.Base.initPage('#accordionGroups');
```

Dans le contrôleur, j'ai dû insérer le code de la section index dans la section destroy afin qu'elle dispose de tous les éléments nécessaires pour afficher l'accordéon, puis le rediriger vers le fichier JavaScript.

Aymeric Moreau

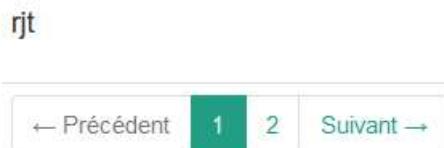
Voici à quoi ressemble mon tableau Trello une fois cette tâche terminé :



J'ai fini toutes les tâches que j'avais à faire. J'ai terminé mon projet.

Je peux maintenant le signaler à mon tuteur de stage et nous effectuerons une vérification rapide ensemble lors d'une visioconférence pour voir s'il y a des choses à changer.

Le premier problème qu'il m'a montré est qu'en bas des tableaux, au niveau du menu de pagination, il y a un trait gris au-dessus du menu :



Ce trait est là car j'ai placé le menu de pagination dans le footer de la table. Pour l'enlever, j'ai dû ajouter l'attribut CSS border-top: none; au menu.

Ensuite, le deuxième problème est que l'alignement des inputs dans le formulaire d'utilisateur n'est pas bon.

* Prénom <input type="text"/>	* Nom <input type="text"/>
Email <input type="text"/>	
Mot de passe <input type="password"/>	Confirmation du mot de passe <input type="password"/>

Le problème vient du fait que pour certains inputs ou groupements d'inputs, je les ai placés dans une div avec la classe col-xs-12 pour m'assurer que le rassemblement des 2 inputs prenne toute la largeur.

```
<div class="row">
  <div class="col-xs-12">
    <%= f.input :name, autofocus: true, wrapper_html: { class: 'col-xs-6' } %>
    <%= f.input :lastname, wrapper_html: { class: 'col-xs-6' } %>
  </div>
</div>
```

Pour le régler, j'ai simplement dû enlever ces div car elles ne servaient en fait à rien.

Le troisième problème est que les 2 boutons de création ne sont pas côté à côté, ils sont l'un au-dessus de l'autre.



J'ai passé beaucoup de temps sur ce petit problème, je n'arrivais pas à comprendre pourquoi ils ne voulaient pas se mettre côté à côté. Le problème vient d'un attribut Bootstrap auquel je n'avais pas fait attention : col-md-offset-10. Cet attribut sert en fait à décaler l'élément vers la droite en utilisant une marge, et c'est à cause de cette dernière que l'élément ne pouvait pas se mettre à côté car la marge prenait toute la place.

Pour régler le problème, j'ai remplacé l'attribut par pull-right, qui décale simplement les éléments sur la droite sans ajouter de marge.

Le quatrième problème est que lorsque l'on va d'une page à une autre avec le menu de pagination du tableau groupe, par exemple, il recharge toute la page et change également la page du menu de pagination de l'utilisateur. Ce problème vient du fait que lorsqu'on exécute le menu, il est configuré pour exécuter le script index.js.erb du dossier où il se trouve, c'est-à-dire 'user', puisque j'ai tout rassemblé dans l'index de 'user'.

Aymeric Moreau

Voici le code du fichier js :

```
$(' #ajax_content').html("<%= escape_javascript(render("index"))%>")
$(' #ajax_content').ready(function(){
  S2C_Gestion.Base.initPage('#ajax_content')
})
```

Le script génère dans ajax_content (qui est la div contenant le tableau 'user' et l'accordéon 'group') le fichier _index, qui affiche les deux éléments en même temps.

```
<%= render partial: 'authentification/gestion/users/user_table' %>
<!-- ===== Partie Group : -->
<%= render partial: 'authentification/gestion/groups/index' %>
```

Il faut donc séparer tout cela.

Pour cela, j'ai déplacé la ligne render de group dans le fichier index.html.erb.

```
<div id="ajax_content">
  <div id="userTable">
    <%= render "index" %>
  </div>
  <%= render partial:"authentification/gestion/groups/index" %>
</div>
```

Maintenant, je dois changer les fichiers js pour qu'ils rendent les bons fichiers au bon endroit. Voici le code des nouveaux index.js :

user :

```
$(' #userTable').html("<%= escape_javascript(render("index"))%>")
$(' #userTable').ready(function(){
  S2C_Gestion.Base.initPage('#userTable')
})
```

group :

```
$('#collapseTable').html("<%= escape_javascript(render(partial: "authentification/gestion/groups/groupsCollapse_table", locals: {groups: @groups}))%>")
$('#collapseTable').ready(function(){
  S2C_Gestion.Base.initPage('#collapseTable')
})
```

Maintenant, ces deux programmes rendent des éléments spécifiques à des endroits spécifiques.

À ce moment-là, j'ai rencontré un problème : le menu de 'group' n'utilisait toujours pas le fichier index.js de 'group', mais toujours celui de 'user'. Pour arranger cela, j'ai dû lui spécifier quel contrôleur utiliser dans ses paramètres :

```
{action: :index, controller: "authentification/gestion/groups"}
```

Ce qui a fini de régler le problème.

Avec ce dernier problème j'ai totalement fini le 1^{er} projet.

Semaine 6

Cette semaine, nous allons commencer le deuxième projet.

Le premier jour, on nous a donc expliqué en quoi il consistait. Nous allons regrouper les pages 'Mentions légales', 'Liens copyright', 'Mode de paiement' et 'Politique de confidentialité' dans la page 'Infos légales & paiement'. En même temps, nous devrons également mettre à jour les interfaces pour qu'elles correspondent au nouveau design.

Voici à quoi ressemblera la page que nous allons réaliser :

ex «RCS» ex «register»

----- Versions dépliées -----

Mentions légales - Textes supplémentaires

ENREGISTRER

Voir fichier séparé

Modal «Ajouter un lien copyright»

ENREGISTRER

FERMER

Onglet actif - Onglet inactif
(Voir pour standardiser avec Julien)

Au clic
Activation/Desactivation
Activation=Ouverture de l'onglet

ACCORDEON MODES DE PAIEMENT
Versions dépliées

Créer règle standard
Pour les textes informatifs/Conseils

Message de confirmation affiché lors de la réservation

Ce texte peut contenir des tags qui seront remplacés par les valeurs correspondantes lorsque le message sera affiché. Vous pouvez utiliser les balises suivantes :
#AMOUNT#: Montant du paiement - **#ORDER#**: Ordre du chèque - **#ADDRESS#**: Adresse postal - **#DEADLINE#**: Délai d'envoi du chèque

CLIQUER SUR L'ICÔNE POUR PREVISUALISER

Merci de votre commande.
Afin de régler celle-ci merci de nous envoyer votre chèque
d'un montant de #AMOUNT# à l'ordre de :

Ordre Agence Démo1
à l'adresse suivante :
8, rue de l'Agence
12345 VILLE-DEMO

Attention : pour confirmation définitive, il est impératif de nous adresser votre règlement
par chèque dans les 48 heures suivant l'enregistrement de votre réservation.

ENREGISTRER

Configurer le mode de paiement : Paybox

N° de site : N° de rang :
Identifiant interne : Version :
Clé secrète commerçant :

Activer le 3DSecure
(compatible avec le 3D Secure V1.
Si votre contrat a été défini pour utiliser 3D Secure V2 , le 3DSecure est obligatoirement activé par la plateforme)

Utiliser la plateforme de test (afin de vérifier les identifiants)

ENREGISTRER

Configurer le mode de paiement : Atos

Identifiant du marchand :

ENREGISTRER

Configurer le mode de paiement : Payzen Lyra

Algorithme de signature en mode Production : Identifiant boutique : Plateforme :
Algorithme de signature en mode Test : Certificat :
 Utiliser la plateforme de test

ENREGISTRER

Configurer le mode de paiement : Mercanet

Merchant ID : Clé secrète : Version de clé :
 Utiliser la plateforme de test

ENREGISTRER

Aymeric Moreau

Pour ce projet ont nous a demandé de réaliser un document semblable a un cahier des charges où nous devons analyser la page a réalisé, la séparer en partie et en fonctionnalité, noter ce qu'on ne connaît pas encore et le style que l'on va utiliser.

Voici le document :

Refonte de la page “Informations légales et paiements”.

Déroulement du projet :

Analyse de l'image de référence :

- Étudier l'image fournie pour comprendre le design et l'interface utilisateur souhaités.
- Identifier les éléments clés tels que la disposition, les couleurs, les polices, les boutons, et les images.

Établissement des exigences techniques :

- Déterminer les modules et éléments nécessaires pour reproduire le design.
- Évaluer si des fonctionnalités supplémentaires sont nécessaires afin de supporter le nouveau design.

Planification du développement :

- Diviser le projet en tâches distinctes et planifier leur réalisation.
- Identifier les dépendances entre les tâches et les ressources nécessaires.

Développement :

- Développer chaque tâche.

Analyse détaillée de la page "Informations légales et paiements" :

La page en question est celle dédiée aux "Informations légales et paiements". Elle constitue une compilation de quatre pages distinctes : Mentions légales, Politique de confidentialité, "Copyright" et "Modes de paiement". Elle se divise en cinq parties principales :

Formulaire des Mentions Légales :

- La première partie contient des champs pour la raison sociale, l'adresse du siège, le numéro de TVA intra, le numéro de téléphone, l'email, le numéro RCS, la ville du RCS, et le capital social.
- La seconde partie comprend des champs pour le prénom, le nom, et la fonction du directeur de la publication.
- Un bouton "Enregistrer" qui occupe toute la largeur de la page sera ajouté, ce dernier n'étant pas présent dans le modèle initial qui nous a été donné pour l'analyse.

Accordéon Textes Supplémentaires des Mentions Légales :

- Contient deux éditeurs de texte de type TinyMCE pour ajouter des textes supplémentaires.

Accordéon Modes de Paiement :

- La description détaillée de cette section sera fournie à part, avec les différents moyens de paiement et leurs formulaires spécifiques.

Accordéon Liens Copyright :

- Présente un tableau listant les liens à afficher.
- Un bouton vert avec un icône "+" permet d'ajouter un nouveau lien, ouvrant un formulaire dans une modale.
- Des boutons "Monter" et "Descendre" (noirs avec des flèches) pour ajuster l'ordre des liens.
- Une section pour modifier, dupliquer ou supprimer les "Autres liens" ajoutés.

Accordéon Politique de Confidentialité :

- Inclut un champ pour le titre et un éditeur de texte TinyMCE pour le contenu de la politique de confidentialité.

L'accordéon concernant les moyens de paiements :

Aymeric Moreau

Lorsque l'accordéon concernant les moyens de paiements sera ouvert, il révélera un système d'onglets représentant les différents moyens de paiement disponibles. En cliquant sur l'un de ces onglets, l'utilisateur pourra accéder au formulaire spécifique aux informations relatives au moyen de paiement sélectionné.

La conception des onglets implique l'utilisation de nuances de gris pour faciliter la navigation. Chaque onglet sera de couleur gris clair par défaut. Toutefois, l'onglet actuellement sélectionné par l'utilisateur devra se distinguer par une teinte nettement plus foncée.

En termes de fonctionnalités, plusieurs boutons seront intégrés. Il y aura un bouton dédié à l'enregistrement des informations saisies dans le formulaire. En outre, un système de bascule sera mis en place pour permettre aux utilisateurs d'activer ou de désactiver les différents moyens de paiement selon leur préférence ou nécessité.

La page de gestion des moyens de paiement comprendra cinq sections distinctes, chacune dédiée à un mode de paiement spécifique. Chaque section nécessitera la création d'un formulaire unique en raison de la variabilité des champs requis pour chaque méthode de paiement. Voici le détail de chaque section :

Paiement par Chèque :

- Le formulaire comprendra trois champs pour saisir l'ordre du chèque, l'adresse postale complète, et le délai maximum autorisé pour l'envoi.
- Une boîte grise contenant des informations sur la réservation sera présente.
- Un éditeur de texte (tiny) permettra de rédiger un message de réservation, accompagné d'un bouton pour prévisualiser ce message.

Paiement via Paybox :

- Les champs requis incluront le numéro du site, le numéro du rang, l'identifiant interne, la version, et la clé secrète du commerçant.
- Deux cases à cocher seront disponibles : l'une pour activer le 3D Secure, et l'autre pour utiliser la plateforme de test.

Paiement via Atos :

- Ce formulaire se limitera à un seul champ pour l'identifiant du marchand.

Paiement via PayZen Lyra :

- Le formulaire comprendra cinq champs : l'algorithme de signature en mode production, l'algorithme de signature en mode test, l'identifiant de la boutique, le certificat, et la plateforme.
- Une case à cocher permettra de sélectionner le mode de test.

Paiement via Mercanet :

- Il y aura trois champs pour l'identifiant du marchand, la clé secrète, et la version de la clé.
- Une case à cocher pour sélectionner le mode de test sera également présente.

Pour chaque moyen de paiement, l'en-tête du formulaire affichera l'information "Configurer le mode de paiement : [Nom du moyen de paiement]" pour une meilleure orientation de l'utilisateur.

Élément inconnu (à cherché ou à créer) :

Render un élément qui vient d'un autre module

La section de prévisualisation dans le mode de paiement chèque

Bouton :

Le bouton ajouter

Le bouton dupliquer

Les bouton de sélection d'un mode de paiement + le possible bouton pour ON/OFF le mode

Indication Style :

Bouton enregistrer :

classe scss : btn-submit

Couleur : Variable `$action_fo` qui correspond à `#097896`;

Pas d'icône

Prend toute la largeur

Texte : de couleur blanche et fait 30px

Bouton Ajouter :

classes scss : btn add

Couleur : Variable `$action_bo` qui correspond à `#229e84`;

icône : `fa-plus` qui correspond à un plus

taille : 36px de hauteur

Aymeric Moreau

Texte : de couleur blanche et fait 18px

Bouton Monter et descendre :

Classe scss : **btn-ordering** qui est à refaire

Couleur : Variable **\$unclassified**: qui correspond à **#000000**;

Icône : fa-angle-up Λ et fa-angle-down V

Taille : 36px de hauteur

Texte : de couleur blanche et fait 18px

Bouton dupliquer :

Classe scss : **btn-duplicate** .

Couleur : Variable **\$action_bo** qui correspond à **#229e84**;

Icône : **fa-copy**

Taille : 36px de hauteur

Texte : de couleur blanche et fait 18px

Bouton modifier :

Classes scss : **btn-edit** .

Couleur : Variable **\$action_bo** qui correspond à **#229e84**;

Icône : **fa-pencil** qui représente un crayon

Taille : 36px de hauteur

Texte : de couleur blanche et fait 18px

Bouton supprimer :

Classes scss : **btn-destroy** .

Couleur : Variable **\$danger** qui correspond à **#b53830**;

Icône : fa-trash-can qui représente une poubelle

Taille : 36px de hauteur

Texte : de couleur blanche et fait 18px

Bouton sélection du mode de paiement :

Classe scss : pas de classe faite

Couleur : Variable **\$base** qui correspond à **#53585b**; quand sélectionner et **#7e858b** quand il ne l'ai pas

Icône : pas d'icône mais un bouton pas encore défini pour l'activer et désactiver

Taille : 36px de hauteur

Texte : de couleur blanche et fait 18px

Aymeric Moreau

Bouton Prévisualiser :

Classe scss : pas de classe faite

Couleur : Variable **\$unclassified**: qui correspond à **#000000**;

Icône : pas d'icône

Taille : 36px de hauteur

Texte : de couleur blanche et fait 30px

L'accordéon :

classe bootstrap : **panel panel-gray no-vertical-margin**.

Couleur : Variable **\$base** qui correspond à **#53585b**;

Texte : de couleur blanc et fait 18px

Il y a donc 3 fonctions scss à créer pour les boutons montés descente, le bouton de sélection du mode de paiement et pour le bouton de prévisualisation.

Et voici notre diagramme de Grantt prévisionnel :

Bleu c'est les tache qui m'on été attribué jaune ce sont c'elles qui ont été attribué a mon collègue et vert c'est les taches en commun.

Première moitié :

	lundi 12-02-2024	mardi 13-02-2024	mercredi 14-02-2024	jeudi 15-02-2024	vendredi 16-02-2024	samedi 17-02-2024	dimanche 18-02-2024
Analyse et préparatifs du projet							
Mise en place de la page/architecture et familiarisation avec les différents modules.		Nécessite d'être finis Mardi afin de ne pas décaler le projet					
Élaboration du formulaire des mentions légales.							
Configuration des accordéons, sans contenu							
Accordéon mentions légales							
Accordéon des liens copyright							
Accordéon Politique de confidentialité							
Mise en place du système d'onglet pour l'accordéon des moyens de paiement							
Formulaire pour Les chèques							
Formulaire pour Paybox							
Formulaire pour Atos							
Formulaire PayZen Lyra							
Formulaire Mercanet							
correction de bug et finalisation							

Aymeric Moreau

2ème moitié :

Tableau Global :

La première tâche du tableau, qui est donc une tâche commune, est la réalisation de ce tableau ainsi que du document d'analyse du projet qui se trouve plus haut.

La deuxième tâche, qui est également commune, a été de réaliser les premières préparations de la page. Cela signifie de décider comment nous allons créer cette page, où la créer et comment nous

Aymeric Moreau

allons rassembler des éléments provenant de plusieurs modules au même endroit. Nous ne pouvons pas avancer tant que nous n'avons pas terminé cette tâche.

Pour commencer, parlons de l'organisation de l'application dans son ensemble. L'application est séparée en différentes 'gems' que nous appelons modules dans l'entreprise. Chaque module correspond à des fonctionnalités différentes ou a des catégories de fonctionnalités qui peuvent être, ou non, intégrées au produit final. Par exemple, si le site commandé est un site vitrine, le module n'a pas nécessairement besoin de toutes les fonctionnalités telles que le module de moyen de paiement.

J'explique cela car dans ce projet, nous avons utilisé trois modules différents. La raison en est que les fonctionnalités de gestion de paiement et autres ne sont pas regroupées dans le même module. Il y a le module de mode de paiement et de mentions légales, qui rassemble les mentions légales, les liens copyright et la politique de confidentialité. Tous les éléments des modules vont être affichés dans le module Gestion, qui va servir de coquille pour la page. Sa seule fonction est de tout rassembler.

Donc, dans le module Gestion, nous avons créé un contrôleur 'info légales' ainsi qu'un dossier pour les vues, avec à l'intérieur un fichier index.html.erb contenant un titre. Dans le contrôleur, nous avons ajouté une fonction pour afficher l'index, et nous avons ajouté une route vers cette même fonction:

```
get 'infolegale', to: 'infolegales#index', as: 'infolegales_index'
```

Avant d'arriver à cette route, nous avons rencontré quelques problèmes, à savoir qu'il ne trouvait pas l'index dans le contrôleur. Le problème venait du nom que nous avions donné à notre contrôleur, qui ne correspondait pas aux normes de Ruby.

Après avoir corrigé cela, nous avons immédiatement essayé d'afficher la page index, mais cela ne fonctionnait pas. Rien ne s'affichait, même pas les menus de navigation. Le problème était que nous n'avions pas créé de layout pour notre page.

Donc, pour créer le layout, il nous a suffi de créer un fichier html.erb avec le même nom que le contrôleur dans le dossier layout du module, ce qui fait automatiquement le lien entre les deux. Ensuite, dans le layout, nous avons juste eu pour l'instant à rendre le layout 'application', car c'est lui qui affiche les menus de navigation.

Après avoir terminé cela, la page s'est affichée correctement.

Après avoir fait cela nous avons commencé à faire a regardé comment nous allions faire pour afficher sur cette page un élément venant d'un autre module. Pour ce faire nous avons décidé d'importer le formulaire des mentions légales.

La première solution que nous avons trouvée était presque correcte.

Elle consiste à créer une fonction dans le contrôleur de gestion pour afficher le fichier désiré :

```
def form_ml
  render partial: 'mentions_legales/gestion/mentions/index'
end
```

Ensuite, nous créons une route vers cette fonction.

Nous ajoutons également un fichier JS dans le module de gestion que nous incluons dans le layout grâce à cette ligne :

```
<% content_for :js do %>
  <%= javascript_include_tag 'gestion/infolegales/application' %>
```

Dans ce fichier JS, nous récupérons la route que nous avons définie et l'exécutons dans la div prévue à cet effet :

```
$.ajax({
  url: '/mon_controller/ma_partial',
  success: function(html) {
    $('#monElementCible').html(html);
  }
});
```

Malheureusement cette première solution n'a pas fonctionné.

La première approche pour résoudre le problème n'a pas fonctionné pour deux raisons principales. Tout d'abord, la fonction fom_ml et la route doivent être placées dans le module correspondant à la page que l'on souhaite récupérer, ici les mentions légales. De plus, il n'est pas correct d'injecter directement le code dans le fichier JavaScript ; il est préférable que le contrôleur appelle un fichier js.erb qui lui va charger de rendre la page dans la div choisie.

Notre tuteur de stage a également souligné plusieurs points d'amélioration, notamment en évitant de spécifier la route directement dans le JavaScript. Il recommande plutôt d'ajouter un attribut data-url à la div et d'utiliser la fonction `Engine.routes.url_helpers` pour dynamiquement trouver le chemin associé à la route.

Après avoir terminé cela, j'ai commencé ma première tâche individuelle, c'est-à-dire m'occuper du formulaire de mentions légales. J'ai l'avantage qu'il est déjà affiché sur la nouvelle page.

Aymeric Moreau

Voici à quoi ressemble la page :

The screenshot shows the Stock2Com software interface with the 'Mentions légales' (Legal Notices) section selected in the sidebar. The main content area displays the 'A' Mentions pied de pages (Footnotes) page. It includes fields for company information like raison social (Sylvain Delmas), capital, registre, RCS, siège social (17, rue de sully - 37000 TOURS), and TVA intercom. There are also fields for the director (Sylvain Delmas) and their title (gérant). Below this is a rich text editor for the footer text, which includes a note about the site being edited by Sylvain Delmas, agent commercial. The bottom section shows the footer text as it will appear on the website, mentioning Sylvain Delmas, RCS number, address, and TVA number.

Aymeric Moreau

Pour la mettre en forme, j'ai dû séparer les inputs en 3 lignes avec Bootstrap et leur attribuer à chacun une taille avec un "col-xs-". J'ai également ajouté les titres des 2 sections, un bouton "Enregistrer" qui prend toute la largeur et enlevé les textareas qui ne font plus partie de ce formulaire.

Code :

```
<%= simple_form_for @mention, :horizontal_form =>
  url: gestion_mention_path(@mention.id),
  wrapper: :horizontal_form, remote: true,
  data: {s2c_formhelper: 'remote,saving_spinner'},
  method: :patch do |f| %>

    <p class="text-center"> <strong> Mentions légales obligatoires</strong></p>

<div class="row"> <%= f.input :raison_social, wrapper_html: { class: 'col-xs-6' } %>
  <%= f.input :tva_intercom, label: "TVA Intra", wrapper_html: { class: 'col-xs-2' } %>
  <%= f.input :rcs, label: "N°RSC", wrapper_html: { class: 'col-xs-2' } %>
  <%= f.input :registre, label: "Ville RCS", wrapper_html: { class: 'col-xs-2' } %>
</div>

<div class="row"> <%= f.input :siege_social, wrapper_html: { class: 'col-xs-6' } %>
  <%= f.input :telephone, label: "N° Tél", as: :tel, wrapper_html: { class: 'col-xs-2' } %>
  <%= f.input :mail, label: "Email", as: :email, wrapper_html: { class: 'col-xs-2' } %>
  <%= f.input :capital, wrapper_html: { class: 'col-xs-2' } %>
</div>

<p class="text-center"> <strong> Directeur de publication </strong></p>

<div class="row"> <%= f.input :directeur_pub, label: "Prénom, Nom", wrapper_html: { class: 'col-xs-6' } %>
  <%= f.input :titre_directeur_pub, label: "fonction", wrapper_html: { class: 'col-xs-6' } %>
</div>

<div class="row">
  <%= button_tag type: :submit, class: 'btn btn-submit col-xs-12' do %>
    </i> <%= t('global.save') %>
  <% end %>
</div>

<% end %>
```

Rendu :

Mentions légales obligatoires

Raison social	rhjicdfnghyrtereft	TVA Intra	FF213(N°RS	12345	Ville RCS	Registr
Siège social	rjdfbed	N° Tél	XX XX	Email	courrie	Capit	100(▾)

Directeur de publication

Prénom, Nom	Mr. X	fonction	Cadre
----------------	-------	----------	-------

*** SAUVEGARDER**

Ensuite, j'ai dû modifier l'action que réalisait le contrôleur lors de la mise à jour, donc de l'exécution du formulaire. Maintenant, lorsqu'il est mis à jour, il renvoie au fichier update.js qui va s'occuper de faire une actualisation ciblée du formulaire.

Après avoir fini ce formulaire, je suis passé au tableau des liens copyright. Voici à quoi ressemblait la page :

The screenshot shows the Stock2Com administrative dashboard. The left sidebar has a 'Liens Copyright' section highlighted. The main content area is titled 'Gestion des liens Copyright (bas de page)'. It features a search bar and a table with three rows:

Texte	Lien	Actions
Mentions légales	/mentions-legales/display	Up/Down
Politique de confidentialité	/politique-confidentialite/display	Up/Down
Conception Stock2Com	https://www.stock2com.com/	Up/Down

J'ai donc commencé par modifier l'index.js du module mention légale de lien copyright.

J'ai juste eu à changer l'endroit où il cible pour viser la div que j'ai créée dans l'accordéon :

```
$('#ajaxModal').modal('hide');
$('#lien').html("<%= escape_javascript(render("index"))%>");
$('#lien').ready(function(){
  S2C_Gestion.Base.initPage('#lien')
  S2C_Utils.ModalAjax.reset();
  $("#lien .ajax-modal").on('am.success', function(){
    S2C_Gestion.Base.initPage('#lien .modal-content')
    S2C_Utils.ModalAjax.initMultiGlobalize();
  });
  new window.SimpleOrderingUI.Base
})
```

Ensuite, dans l'accordéon de l'index de gestion, j'ai ajouté un data-url avec la route de l'index :

```
data-url-confidentialite="<%>= MentionsLegales::Engine.routes.url_helpers.edit_gestion_confidentialite_path(@confidentialite) %>">
```

Pour pouvoir aller chercher l'index.js avec mon javascript :

```
$.ajax({
  url: $('#panel-copyrigh').data('url-copyrigh'),
  dataType: 'script'
});
```

Ensuite, j'ai mis en forme le tableau :

```
<table class="table table-hover table-striped table-condensed table-responsive">
  <thead>
    </thead>
  <tbody >% if @ordering_possible %> class="simple_ordering_sortables" data-simple-ordering-url="<%>= simple_ordering_reorder_gestion_copyright_links_path()%>"<% end %>>
  <% @links.each do |link| %>
    <tr>
      <td><%= link.label %> <%= link.id %></td>
      <td class="fit">
        <div class=" pull-right">
          <% if link.editable %>
            <div style="display: inline-block; vertical-align: text-top;">
              <%= link_to duplicate_copyright_id_gestion_copyright_links_path(id: link.id), class: "btn btn-duplicate", remote: true,
                data: {"toggle" => "tooltip", "placement" => "top", "s2c-remote-links" => 'avoid_change_url'},
                title: t("produits_voyages.gestion.products.index.duplicate") do %>
                <% end %>
              <%= link_to edit_gestion_copyright_link_path(link), class: " ajax-modal btn btn-edit", data: {"toggle" => "tooltip", "placement" => "top"},
                title: t('mentions_legales.copyright_links.index.edit') do %>
                  <!-- ajax-modal -->
                <% end %>
              <%= link_to gestion_copyright_link_path(link), class: 'btn btn-destroy delete-modal ', data: {
                modal_remote_links: true,
                method: "DELETE",
                confirm: t('mentions_legales.copyright_links.index.destroy')} do %>
                <% end %>
              <% end %>
            </div>
          </td>
        <td class="fit">
          <%= render 'simple_ordering/handle_buttons', item_engine: MentionsLegales, up_text: 'Monter', down_text: 'Descendre',
            btn_class: ' btn btn-ordering ', item_path: 'gestion_copyright_link', item_id: link.id, item_first_id: @links.first.id, item_last_id: @links.last.id, mask: !@ordering_possible %>
        </td>
      </tr>
    <% end %>
  </tbody>
</table>
```

J'ai enlevé l'en-tête et la colonne tableau, j'ai ajouté les boutons "Éditer", "Supprimer" et "Dupliquer" pour les liens éditables, j'ai modifié les boutons d'ordre en leur attribuant la classe scss btn-ordering et j'ai modifié les attributs Bootstrap du tableau en enlevant table_bordered.

Voici à quoi ressemble le tableau :

				+ Ajouter
	▲ Monter	▼ Descendre		
Mentions légales				
Conception Stock2Com				
hyfh2452				▲ Monter ▼ Descendre
Politique de confidentialité				▲ Monter ▼ Descendre
test				▲ Monter ▼ Descendre
hdrth				▲ Monter ▼ Descendre

Après cela, j'ai modifié la modale pour qu'elle corresponde au nouveau style. J'ai simplement ajouté les attributs `modal_title`, `modal_footer`, `modal_no_header` aux fonctions qui utilisent une modale

```
def new
  @link = CopyrightLink.new
  @modal = !request.xhr?
  @modal_title = t('mentions_legales.copyright_links.new.title')
  @modal_form = 'new_copyright_link'
  @modal_footer_full_width = true
  @modal_no_header = true
  renderDependingFormat :new
end
```

Rendu :

Ajouter un lien copyright

* Texte

* Lien

ENREGISTRER

FERMER

Ensuite, j'ai mis en place l'actualisation ciblée en JavaScript. Pour cela, j'ai créé un fichier create.js à l'intérieur duquel se trouve un code qui ressemble beaucoup à celui de index.js, mais j'ai simplement ajouté une vérification pour voir si le lien ajouté/modifié est valide ou non.

```
<% if @link.valid? %>
  $('#ajaxModal').modal('hide');
  $('#lien').html("<%= escape_javascript(render("index"))%>")
  $('#lien').ready(function(){
    S2C_Gestion.Base.initPage('#lien')
    S2C_Utils.ModalAjax.reset();
    $("#lien .ajax-modal").on('am.success', function(){
      S2C_Gestion.Base.initPage('#lien .modal-content')
      S2C_Utils.ModalAjax.initMultiGlobalize();
    });
    new window.SimpleOrderingUI.Base
  })
<%else%>
  $('#ajaxModal .modal-body').html("<%= escape_javascript(render(
    partial: "mentions_legales/gestion/copyright_links/form", locals: { link: @link
    , url: gestion_copyright_links_path}))%>")
  S2C_Gestion.Base.initPage('#ajaxModal .modal-body')
  S2C_Utils.ModalAjax.initMultiGlobalize();
<%end%>
```

Si le lien est valide, il réaffiche le tableau en entier. S'il ne l'est pas, il réaffiche le formulaire dans la modale pour que le message d'erreur apparaisse.

Je n'ai pas réussi à accomplir toutes mes tâches cette semaine, il me manque la gestion du bouton "dupliquer".

Semaine 7

C'est la dernière semaine de stage. Il faut que nous fassions notre maximum pour finir ce projet. Les tâches que je dois accomplir sont les suivantes : terminer l'accordéon du lien copyright, mettre en place avec Bastien le système d'onglet pour les modes de paiement, créer le formulaire pour les modes de paiement paybox, payzen et Mercanet, et enfin corriger les erreurs signalées par mon maître de stage.

La première tâche que j'ai réalisée a été de finaliser le bouton de duplication de lien copyright. Ce qui m'a donné le plus de difficultés, c'est de comprendre comment le faire. J'ai commencé comme d'habitude par chercher dans le projet s'il existait une page avec une fonctionnalité similaire.

J'en ai trouvé une dans la page de gestion des produits :



Voici la fonction du contrôleur qui est reliée à ce bouton :

```
##  
# Dupliquer un produit  
#  
def duplicate  
  # TODO: en attente de la mise à jour de l'interface d'édition des produits indiv et groupe  
  #  
  if @product.domain.tag == "bus_travel"  
    new_product = @product.recursive_clone  
    if new_product  
      new_product.synchronize(false, true)  
    end  
  else  
    new_product = ProduitsVoyages::ProductsService.new.duplicate(@product.id)  
  end  
  list_products  
  renderDependingFormat :duplicate  
end
```

Une fois cela découvert, j'ai commencé à créer ma fonction de duplication dans mon contrôleur. J'ai commencé par faire l'actualisation ciblée, comme je suis maintenant habitué à le faire, j'ai repris le même code que celui de create.js.

Ensuite, j'ai créé une route pour ma nouvelle fonction :

```
resources :copyright_links, except: :show do
  collection do
    get 'duplicate/:id', as: 'duplicate_copyright_id' , to: 'copyright_links#duplicate'
  end
  simple_ordering_concern
end
end
```

Puis, j'ai cherché à dupliquer l'objet que j'ai récupéré.

J'ai essayé de reproduire exactement le code des produits, mais cela n'a pas fonctionné ; il ne trouvait pas la fonction "duplicate".

J'ai donc commencé à chercher dans les produits où cette fonction était définie. J'ai regardé dans le modèle, dans le constructeur, dans les helpers, mais la fonction n'était pas dans ce module. J'ai donc demandé à mon maître de stage, Julien, où se trouvait cette fonction. J'ai appris grâce à lui que je ne devais pas utiliser cette fonction, car les produits ont un fonctionnement très spécifique qui ne peut pas être utilisé sur d'autres modules. Il m'a conseillé de me baser sur le "clonage récursif".

Dès lors, j'ai entamé à me renseigner sur ce sujet en cherchant sur internet. Au fil de mes recherches, je me suis rendu compte que ce n'était pas une classe provenant d'une gemme ou d'un framework, mais en fait un module outil que l'entreprise avait créé. "Recursive_clone" était en fait le nom de la fonction que je devais utiliser.

Voici le code de ma fonction :

```
def duplicate
  original_link = CopyrightLink.find(params[:id])

  new_link = original_link.recursive_clone
  new_link.save

  list_links
  renderDependingFormat :duplicate
end
```

Je commence par récupérer le lien correspondant à l'ID passé en paramètre.

Je clone ce lien dans une variable et le sauvegarde.

Ensuite, j'utilise "list_links" pour récupérer tous les liens et je redirige vers "duplicate.js" qui va mettre à jour le tableau.

Mais même avec ce code, il y avait encore un problème : la sauvegarde du nouvel objet ne passait pas.

Grâce à ce code, j'ai pu obtenir un message d'erreur :

```
if new_link.save  
  flash[:success] = 'Link save réussie'  
else  
  flash[:error] = "Original : #{original_link.inspect}- label : #{original_link.label}- Url : #{original_link.url}-  
  --- Nouveau : #{new_link.inspect}- label : #{new_link.label}- Url : #{new_link.url} Failed to save link: #{new_link.errors.full_messages.join(', ')}"  
end
```

Voici le message :

```
Original :  
#<MentionsLegales::CopyrightLink id: 4,  
editable: true, position: 2, created_at:  
"2024-02-15 13:21:23", updated_at:  
"2024-02-19 10:13:05">- label: hyfh245-  
Url: https://www.flfj.com- — Nouveau :  
#<MentionsLegales::CopyrightLink id: nil,  
editable: true, position: 2, created_at: nil,  
updated_at: nil>- label: hyfh245- Url :  
https://www.flfj.com Failed to save link.  
Vous devez remplir au moins une  
traduction complète
```

Tout ce qui est en haut, c'est moi qui ai choisi de l'afficher pour vérifier que la copie se faisait bien. Le message d'erreur est en bas : dit qu'il n'y a pas de traduction pour ce lien.

En fait, les traductions d'un lien ne passent pas par le même modèle ; elles passent par "lien_traduction" et il y a toujours un espace réservé, même s'il est vide. Il faut donc que j'exclue cette partie pour que la sauvegarde fonctionne. Pour faire cela, mon maître de stage m'a donné cet exemple :

```
CLONE_EXCLUDES_FIELDS_TRANSLATIONS = ['id', 'produits_voyages_product_id', 'slug',  
'created_at', 'updated_at']
```

J'ai dû créer ma version avec les attributs du lien copyright et je l'ai mise dans le modèle.

```
CLONE_EXCLUDES_FIELDS_TRANSLATIONS = ['id', 'editable', 'position', 'created_at', 'updated_at', ]
```

Après avoir fait cela, la duplication s'est enfin mise à fonctionner.

Ensuite je suis passé sur les modes de paiement.

Mon collègue Bastien avait déjà commencé à bosser dessus pendant que je terminais mon bouton duplicate. il a réalisé le système d'onglet :

```
<div class="panel-body p-0">
  <div id="paiement-tab" >
    <ul class="nav nav-pills">
      <li class="active">
        <a href="#cheques" data-toggle="tab">Chèques</a>
      </li>
      <li>
        <a href="#paybox" data-toggle="tab">Paybox</a>
      </li>
      <li>
        <a href="#atos" data-toggle="tab">Atos</a>
      </li>
      <li>
        <a href="#payzen_lyra" data-toggle="tab">PayZen Lyra</a>
      </li>
      <li>
        <a href="#mercanet" data-toggle="tab">Mercanet</a>
      </li>
    </ul>
    <div class="tab-content clearfix">
      <div class="tab-pane active" id="cheques">
        <h3>Configurer le mode de paiement pour : Chèque</h3>

      </div>
      <div class="tab-pane" id="paybox">
        <h3>Configurer le mode de paiement pour : Paybox</h3>

      </div>
      <div class="tab-pane" id="atos">
        <h3>Configurer le mode de paiement pour : Atos</h3>

      </div>
      <div class="tab-pane" id="payzen_lyra">
        <h3>Configurer le mode de paiement pour : PayZen Lyra</h3>

      </div>
      <div class="tab-pane" id="mercanet">
        <h3>Configurer le mode de paiement pour : Mercanet</h3>

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

Avant il ressemblait a cela :



Maintenant :



Il était initialement placé dans l'index du module de gestion, mais nous avons finalement décidé de le déplacer dans le module de mode de paiement. Cette décision découle du choix que nous avons fait concernant la méthode que nous allons utiliser. Nous avions deux options : soit rendre chaque formulaire un par un lorsqu'on clique sur leur bouton, soit les rendre tous d'un coup au chargement de la page.

La première option présentait l'avantage d'alléger le chargement initial de la page et de ne charger que les éléments dont nous avons réellement besoin. Cependant, elle présente comme inconvénient le fait que l'affichage du formulaire peut entraîner un délai entre l'exécution du bouton et l'affichage du formulaire, ce qui n'est pas souhaitable et pour les utilisateurs qui passent rapidement entre les onglets, cela peut entraîner plusieurs chargements de contenu et une expérience moins fluide. .

Pour la 2ème solution l'initialisation des formulaires au chargement de la page elle comme problème que s'il y a beaucoup de chose a chargé cela va mettre plus de temps pour afficher la page. Pour les avantages cela facilite la mise en cache, garantie une navigation entre les onglets fluides, cela fait un meilleur référencement, car le navigateur peut indexer tous les contenus des onglets au chargement de la page.

Nous avons choisi la deuxième solution, car les formulaires sont légers, cela ne va donc pas retarder l'affichage de la page. Ainsi, dans notre index de gestion, nous avons cherché l'index du mode de paiement comme nous le faisons pour les autres.

1. On crée un fichier index.js.erb dans le dossier du mode de paiement qui va rendre l'index dans la div prévue à cet effet dans le module gestion.
2. Dans le fichier JavaScript de la gestion, nous allons chercher le fichier index.js grâce à la route récupérée avec la fonction route.helper.

Ensuite, nous avons commencé à nous occuper de la mise en forme des formulaires.

J'ai commencé par modifier le bouton de sauvegarde en ajoutant la classe "btn-submit" pour appliquer le nouveau style et "col-xs-12" pour qu'il prenne toute la largeur :

```
<button class="btn btn-submit col-xs-12" style="margin-top: 1%; ">
  </i> <%= t 'global.save' %>
</button>
```

Ensuite, je suis passé sur le formulaire du mode Payzen :

```
<div role="tabpanel" class="tab-pane fade in" id="payzen">
  <div class="panel">
    <div class="panel-body">
      <div class="paiement <%= @configuration.payzen_state ? '' : 'hide' %>" id="paiement-<%= payment %>">
        <%= simple_form_for @payzen, url: gestion_payzen_path(@payzen), method: :put, html: {autocomplete: 'off', class: 'form-inline'} do |f| %>
          <div class="row">
            <%= f.input :production_mode_algorithm, as: :select, wrapper_html: { class: 'col-xs-6' },
               collection: ::ModesPaiement::Payzen.signature_algorithms.map {|p,value| [t("activerecord.attributes.modes_paiement/payzen.signature_algorithms.#{p}"), value]},
               include_blank: false , :input_html => { :class => 'pull-right' }%>
            <%= f.input :shop_id , wrapper_html: { class: 'col-xs-3' }, :input_html => { :class => 'pull-right' }%>
            <%= f.input :platform_type, as: :select, wrapper_html: { class: 'col-xs-3' },
               collection: ::ModesPaiement::Payzen.platform_types.map {|p,value| [t("activerecord.attributes.modes_paiement/payzen.platform_types.#{p}"), p]},
               include_blank: false , :input_html => { :class => 'pull-right' }%>
          </div>
          <div class="row">
            <%= f.input :test_mode_algorithm, as: :select,wrapper_html: { class: 'col-xs-6' },
               collection: ::ModesPaiement::Payzen.signature_algorithms.map {|p,value| [t("activerecord.attributes.modes_paiement/payzen.signature_algorithms.#{p}"), value]},
               include_blank: false , :input_html => { :class => 'pull-right' }%>
            <%= f.input :certificate , wrapper_html: { class: 'col-xs-3' }, :input_html => { :class => 'pull-right' }%>
            <%= f.input :test_platform , wrapper_html: { class: 'col-xs-3' } %>
          </div>
          <div class="row">
            <div class="form-group center col-xs-12 ">
              <%= render 'save_button' %>
            </div>
          </div>
        <% end %>
      </div>
    </div>
  </div>
</div>
```

J'ai ajouté l'attribut Bootstrap "form-inline" au formulaire pour avoir l'affichage des labels directement à côté des inputs.

Sans form-inline :

Merchant ID

Avec form-inline :

Merchant ID

Ensuite, j'ai ajouté la classe "col-xs-" aux éléments du formulaire pour gérer leur taille et qu'ils soient en ligne. J'ai fini par ajouter la classe "pull-right" aux inputs du formulaire pour qu'ils ne collent pas au label.

Rendu final :

Configurer le mode de paiement pour : PayZen Lyra

Algorithm de signature en mode Production	HMAC-SHA-256	Identifiant boutique	<input type="text"/>	Plateforme	Payzen
Algorithm de signature en mode Test	HMAC-SHA-256	Certificat	<input type="text"/>	<input type="checkbox"/> Utiliser la plateforme de test	
SAUVEGARDER					

Ensuite, j'ai essayé de le faire pour le formulaire de Mercanet. J'ai donc ajouté les mêmes attributs Bootstrap. Le seul changement est que je n'ai eu à faire qu'une row avec l'attribut "col-xs-3" sur chaque input pour les aligner. C'est avec ce formulaire que je me suis rendu compte que l'affichage ne se faisait pas correctement, les labels se remettaient au-dessus de l'input. J'ai cherché longtemps comment faire, je n'ai jamais trouvé comment résoudre ce problème. J'ai donc laissé ça de côté et je me suis mis au formulaire de Paybox, mais je n'allais pas refaire la même chose qu'avec les autres

formulaires. J'ai essayé de trouver une autre façon de faire. J'ai abandonné le "form-inline", j'ai essayé de positionner le label moi-même avec du CSS. Voici le code HTML :

```
<div role="tabpanel" class="tab-pane fade in" id="paybox">
  <div class="panel">
    <div class="panel-body p-0 ">=> @configuration.paybox_state ? '' : 'hide' %> id="paiement-&gt; payment %>">
      <% simple_form_for @paybox, url: gestion_paybox_path(@paybox), html: { class: '' }, method: :put do |f| %>
        <% versions = ::ModesPaiement::Paybox.versions.map { |v,value| [t("activerecord.attributes.modes_paiement/paybox.versions.#{v}"), v] } %>
        <div class="row">
          <div class="col-xs-8">
            <div class="row">
              <%> f.input :pbx_site, wrapper_html: { class: 'col-xs-4' }, input_html: { class: 'pull-right' } %>
              <%> f.input :pbx_rank, wrapper_html: { class: 'col-xs-4' }, input_html: { class: 'pull-right' } %>
            </div>
            <div class="row">
              <%> f.input :pbx_id, wrapper_html: { class: 'col-xs-4' }, input_html: { class: 'pull-right' } %>
              <%> f.input :version, as: :select, collection: versions, include_blank: false, wrapper_html: { class: 'col-xs-4' }, input_html: { class: 'pull-right' } %>
            </div>
            <div class="row">
              <%> f.input :key, wrapper_html: { class: 'col-xs-12' }, input_html: { class: 'pull-right' } %>
            </div>
          </div>
          <div class="col-xs-4">
            <div class="row">
              <%> f.input :enable_3ds %>
            </div>
            <div class="row">
              <%> f.input :test_mode %>
            </div>
          </div>
        </div>
        <div class="form-group center col-xs-12">
          <%= render 'save_button' %>
        </div>
      <% end %>
    </div>
  </div>
```

Et le code css :

```
#paiement-paybox .form-group {
  margin-right: 10px;
  margin-bottom: 10px;
  display: flex;
  align-items: center;
}

#paiement-paybox .form-group label {
  margin-right: 5px;
  max-width: 800px;
  width: 100%
}
```

Donc, dans le HTML, j'ai séparé le formulaire en deux colonnes, une de taille 8 pour les inputs de texte et une deuxième de taille 4 pour les deux boutons radio. Ensuite, dans la première colonne, j'ai séparé les inputs en trois lignes comme dans le modèle et en deux lignes pour les boutons radio. J'ai laissé les "col-xs" et les "pull-right" pour gérer leur taille et leur placement par rapport au texte.

Ensuite, dans le CSS, j'applique au "form-group" la propriété "flex" pour pouvoir ensuite les aligner ensemble avec "align-center". Sur les labels, j'ai modifié la taille qu'ils prenaient pour que l'affichage ne se fasse pas sur deux lignes mais bien sur une. Voici le rendu du formulaire :

Configurer le mode de paiement pour : Paybox

N° de site	<input type="text"/>	N° de rang	<input type="text"/>	<input checked="" type="checkbox"/> Activer le 3DSecure (compatible avec le 3D Secure V1. Si votre contrat a été défini pour utiliser 3D Secure V2, le 3DSecure est obligatoirement activé par la plateforme)
Identifiant interne	<input type="text"/>	Version	HMAC	<input checked="" type="checkbox"/> Utiliser la plateforme de test (afin de vérifier les identifiants)
La clé secrète commerçant	<input type="text"/>			SAUVEGARDER

Ce n'est pas parfait, mais il y a moins de problèmes d'affichage sur les écrans plus petits.

C'est par contre la dernière chose que j'ai pu faire durant ce stage. Avec Bastien, nous avons pratiquement réussi à finir ce second projet. Les seules tâches qui restent, c'est de finir la présentation des formulaires et de faire la fonctionnalité de pouvoir activer/désactiver un mode de paiement. Nous avions regardé rapidement avant de passer au formulaire, mais nous n'arrivions vraiment pas à comprendre le JavaScript qui était utilisé, nous l'avions donc mis de côté.

Le dernier jour nous avons eu entretien bilan un avec le président de l'entreprise sur comment c'était passé notre stage et un avec notre tuteur de stage pour le 2ème projet. C'est durant ce 2ème entretien que j'ai appris quand faite une partie de mes problèmes venait enfin du pull-right. Cette classe bootstrap attribut aux éléments la propriété float et c'est à cause de cette propriété que je n'arrivais pas à faire ce que je voulais.

Bilan du stage

Les objectifs :

Nous avons réussi à finir notre 1^{er} projet mais pas notre 2^{ème} projet il nous manquait un peu de temps.

Les écarts sur les plannings sont dus à la sous-estimation de la complexité de certaines tâches et à un temps nécessaire pour résoudre les problèmes ; pour déboguer qui s'avère trop long.

Ce que m'a apporté mon stage :

J'ai pu améliorer mes compétences en javascript et en bootstrap.

J'ai pu découvrir le framework ruby on rails et le scss.

J'ai pu découvrir comment était le métier de développeur web front-end.

J'ai gagné en autonomie.

J'ai vu comment fonctionner une petite entreprise de développement web.

Conclusion

J'ai apprécié ce stage j'ai appris pas mal de chose une bonne condition de travail.

Je suis un peu déçu de ne pas avoir réussi à finir le 2ème projet je suis sûr qu'avec 1 ou 2 jours de plus on aurait pu le terminer.

La seule chose que je trouve dommage durant ce stage c'est qu'on ait fait que de la refonte graphique. J'aurais aimé pouvoir faire plus de code orienté back-end.

Mais ce fut une bonne expérience.