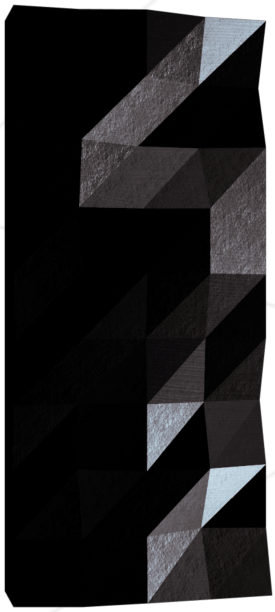

Formation Ruby on Rails - Rush 01

English version coming soon !

Résumé: Ceci est le deuxieme projet complexe (tout est relatif) que vous avez à faire.

Table des matières

Chapitre I

Consignes

- Seule cette page servira de référence : ne vous fiez pas aux bruits de couloir.
- Le sujet peut changer jusqu'à une heure avant le rendu.
- Si aucune information contraire n'est explicitement présente, vous devez assumer les versions de langages suivantes :
  - `Ruby >= 2.3.0`
  - pour `d09 Rails > 5`
  - mais pour tous les autres jours `Rails 4.2.7`
  - `HTML 5`
  - `CSS 3`
- Nous vous interdisons FORMELLEMENT d'utiliser les mots clés `while, for, redo, break, retry, loop et until` dans les codes sources Ruby que vous rendrez. Toute utilisation de ces mots clés est considérée comme triche (et/ou impropre), vous donnant la note de -42.
- Les exercices sont très précisément ordonnés du plus simple au plus complexe. En aucun cas, nous ne porterons attention ni ne prendrons en compte un exercice complexe si un exercice plus simple n'est pas parfaitement réussi.
- Attention aux droits de vos fichiers et de vos répertoires.
- Vous devez suivre la procédure de rendu pour tous vos exercices : seul le travail présent sur votre dépot GIT sera évalué en soutenance.
- Vos exercices seront évalués par vos camarades de piscine.
- Vous ne devez laisser dans votre répertoire aucun autre fichier que ceux explicitement specifiés par les énoncés des exercices.
- Vous avez une question ? Demandez à votre voisin de droite. Sinon, essayez avec votre voisin de gauche.
- Votre manuel de référence s'appelle `Google / man / Internet / ...`.
- Pensez à discuter sur le forum Piscine de votre Intra, ou Slack, ou IRC...

- Lisez attentivement les exemples. Ils pourraient bien requérir des choses qui ne sont pas autrement précisées dans le sujet...
- Par pitié, par Thor et par Odin ! Réfléchissez nom d'une pipe !

Chapitre II

Règles spécifiques de la journée

-
- En dehors des gems de rails vous n'avez droit qu'á ces Gems (bonus compris) :
  - Pdf generation :
    - `Prawn`
    - `Wicked pdf`
    - `Pdfkit`
    - `Wkhtmltopdf`
  - Registration :
    - `Devise`
    - `OmniAuth`
    - `Authlogic`
  - Styling :
    - `Bootstrap-sass`
    - `Rails-bootstrap-forms`
  - Ajax :
    - `Best in place`
  - Rich text :
    - `wysiwyg-rails`
  - CSV :
    - `roo`
  - Charts :
    - `LazyHighCharts`
  - Date range :
    - `jquery datepicker`

Chapitre III

Préambule

`Usefull Facts`
Google runs on 5000 times more code than the original space shuttle

The microcontroller inside a Macbook charger is about as powerful as the original Macintosh computer.

"The quick brown fox jumps over the lazy dog" is an English-language pangram, a phrase that contains all of the letters of the alphabet.

If you eat a teaspoon of sugar after eating something spicy, it will completely neutralize the heat.

A day on Venus is longer than a year on Venus.

The term “nerd” originated from Dr. Seuss’ 1950’s book, If I Ran The Zoo.”

Scotland’s national animal is the unicorn

In ancient Rome, when a man testified in court he would swear on his testicles.

A donkey will sink in quicksand but a mule won’t.

In the 40’s, the Bich pen was changed to Bic for fear that Americans would pronounce it’Bitch.’

A woodchuck breathes only 10 times in hibernation.

A cat’s jaw cannot move sideways.

An ostrich’s eye is bigger than its brain.

If you put a drop of liquor on a scorpion, it will instantly go mad and sting itself to death.

The average person falls asleep in seven minutes.

The Declaration of Independence was written on hemp (marijuana) paper.

A group of kangaroos is called a mob.

IBM's motto is 'Think'. Apple later made their motto 'Think different'.

Albert Einstein and Charles Darwin both married their first cousins.

Charlie Chaplin once won third prize in a Charlie Chaplin look-alike contest.

Most dust particles in your house are made from dead skin !

In the "May I have a large container of coffea" sentence, the number of letters in each words makes the (begining of) the pi number.

It would take approximately 31.7 years to count off 1 billion seconds.

Bikinis and tampons were invented by men.

Kissing is more hygienic and healthier than shaking hands.

It is impossible for a human to sneeze with your eyes open.

Chapitre IV

Partie obligatoire

Vous voici arrivés à ce deuxième rush qui clôturera ces deux semaines de piscine.

C'est l'occasion pour vous de démontrer vos compétences avec Rails, de prouver votre extraordinaire efficacite de statup-o-freelanc-o-wak-o-developeur.

Les 'vues' ne sont pas imposées, c'est à vous de definir si tout est sur une page ou s'il n'y a qu'un mot par page. Cependant, la mise en page est requise ainsi qu'un minimum de structure :

- `Incitation` : conduit l'utilisateur à effectuer des actions spécifiques.
- `Distinction` : groupe des fonctions de même type facilitant l'accès, la mémorisation et l'apprentissage.
- `Lisibilité` : éclaircir et limiter le nombre d'éléments différents présents sur l'interface
- `Concision` :afficher le contenu de façon à faciliter l'exploitation des données

Ce ne sont pas la des regles du rush, mais une incitation au bon sens qui facilitera, aussi, votre correction.

Vous n'avez pas à faire de tests. J'entends par là : cela ne sera pas évalué, mais on n'est jamais a l'abri des effets de bord du développement d'une feature.

Vous `devez` par contre avoir une seed qui suffit à démontrer le bon fonctionement de chaque fonctionalité.

Il est aussi conseillé de lire tout le sujet avant de commencer ;) .

IV.1 Iteration 1 : Fonctionalité Non Commerciales

Vous devez commencer par réaliser un projet de base comportant plusieurs parties, chacunes rassemblées dans un sprint :

IV.1.1 Sprint1 : User accounts

Vous devez mettre en place un sytème d'authentification, avec comme feature obligatoire :

- Pas de password en clair (en logs ou en DB)
- Formulaire de sign-in // sign-out
- Affichage :
  - Connexion
  - Systeme de notification flash des évènements
  - Possibilités pour les utilisateurs de modifier leurs infos

Ainsi qu'une gestion de priviléges relatives aux types de comptes :

- `admin` : a son back-office propre avec un contrôle total de `tout` le CRUD.
- `operator` : fonctionalités basiques (accès à ses données personnelles), et doit appartenir à un des groupes suivants (il ne peut appartenir qu'a un seul) :
  - `Production`
  - `International`
  - `Commercial`
- `manager` : fonctionalités basiques (accès à ses données personnelles), accès aux back-offices de chaques sprints (sauf celui de l'admin), accès en lecture aux données personelles (toutes), et assignation des users aux groupes.
- `non enregistre` : accès seulement aux parties publiques.

`Rappel: Vous n'avez pas la permission d'utiliser des gems qui feront le taff a votre place, genre 'rolify', 'can-can' ...`

IV.1.2 Sprint 2 : Admin back-office

Comme il était dit plus haut, il doit y avoir tout le `CRUD` de toute l'application représenté ici, tout ce que vous allez développer au long des exercices réalises aujourd'hui devra être ajouté ici et en libre accès à des fins d'administration.

Seul lui pourra :

- assigner / modifier des privilèges à des users.
- créer/modifier un 'user account' et en supprimer.

`L'admin est -tout-puissant- sur les données, mais ne doit pas pour autant créer d'erreurs : 'stored nil values', 'non uniques IDS' ou 'unvalid data'`

`Rappel: Vous n'avez pas la permission d'utiliser des gems type 'rails-admin', 'Active-admin'...`

IV.1.3 Sprint 3 : Site vitrine

Une page d'accueil permettant de faire une presentation de votre entreprise fictive.
Cette partie est publique
Cette page est de votre responsabilite et doit comporter `obligatoirement` :

- des liens vers : s'enregistrer, se connecter.
- des photos, une (ou plus) video(s)

Back-office :

- Aucun

IV.1.4 Sprint 4 : In mail

Dois-je vous expliquer le concept du 'in mail' ? Bon, dans le doute, je liste au moins les features obligatoires :

- `in box` : pour tous les users enregistrés avec tout les in-mail à la destinatiion de l'user connecté, et mention de statut : 'lu ou nouveau'.
- `out box` : pour tous les users enregistrés avec tout les in-mails emvoyés par l'user connecté. Avec mention 'lu' si le destinataire l'a ouvert.
- `send` : tout le monde peut envoyer des 'in mails' individuelement.
- `contact list` : liste de tous les users enregistrés avec liens vers "in mail cette personne" (non ca j'explique pas), et mention du groupe auquel il appartient
- `pdf` : les in-mail recus / envoyes ont un bouton qui génère un pdf avec le contenu du mail (passer par la fonction d'impression du navigateur ne sera pas admise, allons... vous vous en doutiez.).

Back-office :

- Envoi de in-mail aux groupes.
- Envoi de mail à tous.
- historique d'activité, liste des envois avec mention 'lu' si ouvert par le destinataire, à la facon d'un log [date/heure][titre][expediteur][destinataire][link to content] :
  - Pour tous
  - Par users
  - Par groupes
- De même, un manager peut via l'historique lire tous les in-mails (meuu non c'est pas du tout flippant, Mr Tuttle)

IV.2 Iteration 2 : Fonctionalités Commerciales

Apres avoir mis en place une infrastructure de base, vous devez maintenant vous occuper de la partie coeur de metier de votre entreprise fictive.

IV.2.1 Sprint 1 : Client Database

Comme son nom l'indique, c'est une liste de clients et elle va se prototyper comme suit :

nom, prenom, email, tel, company, fonction

Elle devra être accessible, par tout utilisateur enregistré et lisible par ordre alphabetique et par company. On doit aussi pouvoir en importer des nouveaux, via CSV grâce à la gem roo.

Back-office :

- export en CSV
- Historique d'activité : par company et par client, on doit voir les actions qui concernent ces entités. (dans l'etat actuel du code c'est irrelevant, mais avec les features du prochain sprint, ca prendra tout son sens). Il devra y figurer entre-autres :
  - Imports, date et par qui
  - Quotes, date et par qui
  - Orders, date et par qui
  - Invoices, date et par qui
- suppression d'entrées
- assignation de clients à des opérateurs (tout groupes confondus)

IV.2.2 Sprint 2 : Project Suite

Un projet a :

- un nom, un client et une ou plusieurs quotes (devis, proposition faite à un client par un opérateur)
- une order (commande crée par un opérateur à la demande du client)
- une ou plusieurs in-going invoices(facture de frais de fonctionement, marchandises)
- une out-going invoice (facturation finale du client, concordant avec le bon order)

Chacun de ces elements dispose de :

- Une entete où figure :
  - un client
  - son entreprise
  - son opérateur
  - une date de création
- des items qui ont :
  - un prix
  - une description
- une intro en 'rich text'
- un total en euros '€'

Les opérateurs peuvent créer des invoices, orders et quotes qui `doivent` appartenir à un projet.

`vous etes exemptés du concept de TVA, me remerciez pas`

Back-office :
Vous allez calculer toujours sur une granularite de 1 semaine (axe des abcisses) des montants (axe des ordonnées), et ce sur des plages temporelles sélectionnables via le plugin 'pick-a-date' en tête de page. Il y a trois pages : par `client`, par `entreprise` et pour `tous`, où figurent les deux graphs suivants :

- Somme des recettes (out-going invoices) diminués des depenses (in-going invoices)
- Somme des orders

Seul le manager peut créer un projet.

IV.2.3 Sprint 3 : Survey

Chaque operateur peut créer `un` sondage, les sondages sont publics et vous devez enregistrer les mails (syntaxiquement valides) des sondés non enregistrés.

Les sondages comportent :

- Un titre
- Une intro en'rich-text'
- des questions a reponse oui/non
- un message de remerciement en'rich- text'.

Back-office :

- création de multiples sondages
- validation (mise en public/visible) des sondages
- import par CSV de nouveau sondages
- export par CSV des resultat de sondages
- listing et export par CSV des mails collectés

IV.3 Bonus

Vous vous doutez certainement que les bonus ne seront evalués `que` si tout le reste fonctione parfaitement.
Beaucoup de 'smart-features' ont été omises dans le sujet, trouvez des choses cohérentes et utiles à ajouter pour le peu qu'elles n'utilisent `aucune` gem supplémentaire.

`UN` bonus est une amélioration d'`UN` sprint et de la mise en page, c'est du serieux pas du bonus.