Dorian COFFINET
25 ans
dorian.coffinet@gmail.com

Compte rendu de la création de l’application rails

Table des matières

I. Environnement de développement

1) Version de Ruby

L'application a été codé avec la version 1.9.3-p448 géré avec rvm en version 1.22.14.

2) Gems utilisées

Tout au long du développement de l'application plusieurs gems ont été requise pour l'ajout de fonctionnalités.

Le suivit du tutoriel a été respecté et toutes les fonctionnalités sont disponibles. J'ai du faire quelques manipulations pour faire fonctionner ces fonctionnalités (annotate, spork etc…). Pour annotate j'ai suivit les explications données ici :

http://techhamlet.com/2011/12/how-to-setup-the-latest-rails-on-linux-with-spork-and-autotest/

Néanmoins j'ai utilisé la version 3.2.11 de la gem rails. Par rapport au tutoriel le css, les images et le javascript ne se trouve plus dans /app/public mais dans /app/assets, c'est un des changements.

Les principales gems utilisées en plus sont :

- La gem paperclip pour la sauvegarde des pdfs des utilisateurs en base.
- La gem prawn, prawn_rails et prawn-layout pour la générations de pdfs à la volé de la liste des utilisateurs.
- La gem imc, pour m'entraîner sur quelque chose de nouveau j'ai décidé de créer une gem calculant l'imc d'un utilisateur. C'est une gem très simple mais qui m'a permit de découvrir comment créer et construire une gem (fichier gemspec, tests, rdoc…). Je ne l'ai pas mit à disposition dans les dépots car c'est une gem très simpliste et qui n'a que l'intérêt de comprendre la conception d'une gem. Les tests sont réalisés avec la gem shoulda.

Le Gemfile contient les gems nécessaires à l'installation de l'application.

3) Logiciel de version

Tout au long du développement de l'application j'ai utilisé git (version 1.8.1.2) ainsi que github. Il est essentiel lors d'un développement tel que celui-ci d'avoir un logiciel de version.

II. Améliorations et points importants

1) Améliorations

En plus des options obligatoires données par les consignes, j'ai ajouté :

- Une amélioration de la vue d'un user, affichage d'une miniature pdf si l'utilisateur à ajouté un cv en pdf. Grâce à la gem paperclip.
- Téléchargement à la volé possible de la liste des utilisateurs en pdf. Cette option est disponible dans la vue de la liste des utilisateurs. (grâce à la gem prawn)
- Ajout d'une page sportif qui affiche la liste des utilisateurs non sportifs qui décident de faire du sport.
- Ajout d'une page statistiques qui affiches les différents graphiques (répartitions des imc, boxplots des poids actuels/idéals…). Ces graphiques sont réalisés grâce à highcharts (http://www.highcharts.com/).
- Masquer le choix "Souhaitez faire du sport ?" si le choix "Pratiquez vous un sport ?" est sur vrai en jQuery.
- Création d'un fichier seed.rb qui permet de remplir la base de donnée de plusieurs utilisateurs avec la commande "rake db:seed".
- Passage en scss du code css lié à la vue d'un user. Le scss est souvent utilisé dans les applications rails.
- Mise en place de l'i18n (internationalisation) pour avoir une application en français avec une possibilité de la passer en anglais par la suite.

2) Points importants

Comme cité plus haut deux graphiques sont disponibles dans l'application. Pour les réalisés j'ai réorganisé mon code de la façon suivante :

1. Création d'une classe personnalisée UserStatistics => app/lib/user_statistics.rb. Cette classe me permet de formater les données des users. Cette classe est appelée dans le constructeur des users pour pouvoir être utilisée.
2. Création de partiels pour les graphiques. Un partiel représente un graphique. Ce découpage permet de pouvoir réutiliser les graphiques autre que dans la page sattistiques.

III. Structure de la base de données

La base de données a été amélioré au fur et à mesure de l'avancement du projet. Vous pouvez voir les différentes migrations dans le projets.

Voici un listing de la table User :

id: integer
nom: string
email: string
created_at: datetime
updated_at: datetime
dateNaissance: date
poidsActuel: integer
poidsIdeal: integer
estSportif: boolean
souhaitePratiquerSport: boolean
taille: integer
cvpdf_file_name: string
cvpdf_content_type: string
cvpdf_file_size: integer
cvpdf_updated_at: datetime

Certains champs sont créés automatiquement avec la gems paperclip.

IV. Tests rspec

Les specs sont classées comme suit :

- Specs des controlleurs
    - pages_controller_spec : permet de tester si les pages attendues sont les bonnes ainsi que le titre de chaque pages.
    - user_controller_spec : permet de tester les différentes pages d'un user (new, show, index …). Par exemple pour la page new d'un user les specs vont tester si les champs d'inscription sont bien présents dans le formulaire.
- Specs des models : user_spec permet de tester l'intégrité des données. Par exemple la création d'un user, l'exigence et le format de certain champs…
- Specs des requests
    - layout_links_spec : permet de tester si les liens de l'application nous renvoie vers les bonnes pages.
    - users_spec : permet de tester une inscription réussi ou non à l'aide du formulaire.

Pour pouvoir tester l'ajout de pdf d'un utilisateur, j'ai créé un répertoire contenant les cv exemples : app/spec/fixtures/files/.

Le fichier factories.rb me permet de créer un utilisateur pour les specs. Il est notamment utilisé dans les specs user_controller_spec.

V. Installation de l'application

Voir le fichier README.txt à la racine du tar.gz

VI. Conclusion

Connaissant déjà Ruby et RoR, en l'apprenant de manière autodidacte et par le biais d'un stage l'année dernière. Je n'ai pas rencontré de gros problème. J'ai eu quelques soucis au niveau de la compatibilité de certaines gem en suivant le tutoriel donné en consigne.

Après avoir fini le chapitre 6 je me suis concentré sur les améliorations à faire. Suivre un tutoriel c'est bien mais ce n'est que du copier/coller! J'ai donc mis en pratique ce que j'ai appris durant mon stage de L3pro :

- le découpage du code (utilisation de partiels)
- faire un code réutilisable et clair (refactoring du code)
- le scss
- l'utilisation d'highcharts pour les graphiques

Avec Highcharts j'ai eu quelques soucis pour créer les box-plots de poids, je ne sais pas pourquoi je suis obligé de mettre en dur un lien vers un fichier de highcharts sur le web ligne 1 du fichier views/shared/_graphics_box_plot_poids.html.erb. Je voulais mettre ce fichier dans les asset directement mais je n'ai pas trouvé la solution.

En stage je n'avais pas vraiment vu la création de specs pour une application. Ici j'ai pu créer les specs de A à Z et voir comment cela fonctionne réellement sur différents types des specs.

La création du pdf à la volé et la création de gem en ruby m'étaient étrangère, ce projet a été une bonne occasion de les réaliser.

Ce projet m'a permit d'appliquer mes connaissances acquises via mon stage et de découvrir de nouvelles choses en ruby/rails.