Тестовое задание (Task Manager)

Реализовать функционал Task Manager в соответствии с требованиями к приложению и требованиям к коду (ruby и rails style guide [1][2]).

Требования к тестовому заданию взяты из реальных проектов. Выполнение задания позволит ознакомиться с различными инструментами и гемами, которые используются в реальных проектах, подходами к построению архитектуры приложения.

Требования к реализации:

1. Сделать главную страницу приложения, на которой выводится список всех задач в системе с указанием идентификатора задачи, времени создания, названия и пользователя, на которого назначена задача.
2. Сделать страницу авторизации пользователя в систему Task Manager.
3. После авторизации пользователь попадает на страницу списка своих задач (личный кабинет). В этом списке выводятся задачи со следующими атрибутами: идентификатор задачи, название, описание, статус задачи, время создания.
4. Сделать возможность добавления, редактирования и удаления задач пользователем из списка задач в личном кабинете.
5. Если пользователь - admin, то в списке задач в личном кабинете он видит все задачи в системе. При этом в таблице списка задач указывается ещё пользователь, на которого назначена задача. Он может назначать, редактировать и удалять задачи других пользователей.
6. Реализовать прикрепления файла к задаче. Использовать гем [carrierwave](https://github.com/carrierwaveuploader/carrierwave).
7. Сделать страницу просмотра задачи в личном кабинете. На странице выводится информация о задаче: идентификатор задачи, название, описание задачи, время создания. Если к задаче была прикреплена картинка, то отобразить картинку, иначе отобразить ссылку на скачивание файла. Если пользователь - admin, то выводить пользователя, на которого назначена задача.
8. Реализовать смену состояния задачи отдельными переключателями в списке задач.

Требования к бизнес-логике

1. Модель User. Атрибуты - email (уникальное поле), password, role (роль пользователя - admin, user, можно отдельным полем(атрибутом)).
2. Модель Task. Атрибуты - name, description, user, state. Возможные значения state - new, started, finished. Задача не может существовать без имени.

 Реализовать связь one-to-many между User и Task. Обеспечить целостность данных при удалении объектов (задача не может существовать без пользователя).

 Для реализации state machine использовать гем [state\_machines-activerecord](https://github.com/state-machines/state_machines-activerecord) или [aasm](https://github.com/aasm/aasm).

Требования к контроллерам

1. Реализовать иерархию контроллеров [3]
2. Не использовать scaffold.
3. Реализовать кастомную авторизацию, не использовать гем devise.
4. REST.

Требования к вьюхам

1. Использовать гем [simple\_form](https://github.com/plataformatec/simple_form) для реализации форм.
2. Шаблонизатор - haml/slim.
3. Twitter Bootstrap 3 для вёрстки.

Требования к тестированию

1. TDD.
2. Код покрыть функциональными тестами.
3. Тестовое покрытие кода >92 %. Проверка через гем [simplecov](https://github.com/colszowka/simplecov).

Требования к развёртыванию

1. Код разместить на [github](https://github.com/).
2. Прикрутить [travis ci](https://travis-ci.org/).
3. При наличие собственного внешнего сервера выложить приложение на сервер и прикрутить capistrano для деплоя. При отсутствии - выложить на [Heroku](https://www.heroku.com/).
4. Написать rake-таск для создания фейковых данных. Использовать гем [faker](https://github.com/stympy/faker).
5. Создать учётку admin и user и поместить в seeds.

Оценка задания

 При оценке тестового задания будут учитываться следующие пункты:

1. Выполнение требований к заданию.
2. Следование style guides.
3. Следование требованиям к тестированию, полнота тестов.
4. DRY.

Справочная информация

[1]. Ruby Style Guide - <https://github.com/arbox/ruby-style-guide/blob/master/README-ruRU.md>.

[2]. Rails Style Guide - <https://github.com/arbox/rails-style-guide/blob/master/README-ruRU.md>

[3]. Иерархия контроллеров - <http://habrahabr.ru/post/136461/>

[4]. Rusrails: Ruby on Rails по-русски - <http://rusrails.ru/>