Trabajo 3: Ruby on Rails

Low Travel

Diego Sáinz de Medrano

1. Resumen de la implementación.

Low Travel es una aplicación de blogs escrita en Ruby on Rails. Utiliza una variedad de gemas y modelos para su funcionamiento. A grandes rasgos, el esquema básico de la aplicación es el siguiente:

- User: el usuario tiene un modelo muy sencillo, incluyendo nombre, email, fecha de nacimiento, espacio para una biografía, una url para la web y contraseña. El registro en Low Travel es público, y se ha implementado en la página con la gema “devise”, apta para manejar eficientemente sesiones, registros y cambios de contraseña. Además, el registro es necesario para algunas acciones relacionadas con los viajes.

- Travel: el modelo del post. Este incluye una imagen de “fachada” o portada, un título, un texto de contenido, y etiquetas sobre las que se puede buscar. Está relacionado con el modelo de usuario con una relación “one to many”. El usuario dueño del viaje tiene acceso a su página de edición desde la cual puede realizar cambios y actualizar el estado de público u oculto (la diferencia es que a los viajes ocultos sólo se puede acceder desde la página del usuario que es dueño).

También se incluye un controlador que se utiliza para el manejo de la página de inicio de la web, PagesController. En él está definida la función “home”, cuya vista es la ruta raíz de la aplicación.

Elementos obligatorios:

- Validaciones
  - En User: requerido que el campo del nombre sea diferente de los existentes, una dirección de email con formato correcto (validación interna de “devise”) y que la fecha de nacimiento sea antes de hace 18 años
  - En Travel: requerido que el título sea único y entre 4 y 40 caracteres, el contenido entre 150 y 7000, que al menos tenga una etiqueta y que la imagen de portada sea menor que 5Mb.
- Scopes
  - Todos son para Travel (no existe un índice público de usuarios).
  - `most_recent`: ordena por orden de publicación
  - `published`: filtra los artículos no publicados
  - `default_scope`: los dos anteriores concatenados
  - `with_title (title)`: artículos con `title` incluído en el título
  - `by_year_and_month (year,month)`: artículos con estos datos en la fecha de publicación [deprecado por el uso del siguiente scope]
  - `by_months_ago (months)`: artículos publicados hace `months` meses

2. Elementos opcionales

- El diseño en forma de “grid” mediante el uso de Bootstrap (con la gema “bootstrap-sass” y la importación de los módulos de javascript: esto es debido a que existía un bug en el módulo de Affix que se parcheó antes de incluirlo).
https://github.com/twbs/bootstrap-sass
- El contenido estilizado en Markdown, que se renderiza vía la gema “redcarpet”.
https://github.com/vmg/redcarpet
- La validación de la fecha de nacimiento para mayores de 18 años con la gema “validates-timeliness”.
https://github.com/adzap/validates_timeliness
- La implementación de un sistema de etiquetas con la gema ”acts-as-taggable-on”: permite añadir etiquetas con un campo de texto simple separadas por comas, y con una añadidura a las rutas, permiten encontrar viajes etiquetados.
https://github.com/mbleigh/acts-as-taggable-on
- El uso de la gema ”autoprefixer-rails” para añadir los prefijos que dan mayor compatibilidad de CSS3 con los navegadores que aún no cumplen con los estándares.
https://github.com/ai/autoprefixer-rails
- Para validar el tamaño de los archivos subidos, el uso de la gema ”file_validators”.
https://github.com/musaffa/file_validators
- El uso de rutas más amigables para el usuario con la gema ”friendly_id”. Se usan los nombres de los usuarios y los títulos de los artículos para la generación de urls.
https://github.com/norman/friendly_id

En el repositorio git (https://bitbucket.org/DiegoSd/LowTravel) se puede observar el camino del desarrollo en las diferentes ramas.

El archivo de viajes es la razón de la existencia del scope by_months_ago, y es un parcial que hace uso de una variable de instancia que se genera en el controlador de la Aplicación (es decir, se genera en todos los controladores por herencia) que incluye referencias a relaciones de ActiveRecord conteniendo los correspondientes viajes publicados.

Para evitar la repetición de código, se utilizaron además otros parciales para la renderización de la cabecera de la página, la barra de navegación, los formularios de inscripción y editado de los viajes y la vista compactada (indexed) y completa (travel) de los viajes.

3. Referencias

Demasiadas para haberlas ido coleccionando. En los mensajes de los commits se indican algunas especialmente relevantes para puntos clave del desarrollo, pero los sitios más visitados fueron StackOverflow, RailsCasts, YouTube y en menor medida las Rails Guides (mucho del contenido en estas era demasiado técnico para mi conocimento de Ruby on Rails).

4. Modificaciones respecto a trabajos anteriores

Al desarrollar la aplicación solo, teniendo que pelearme con mi desconocimento del lenguaje, hay algunas cosas aún por hacer que estaban incluidas en el proyecto original, por ejemplo la inclusión de enlaces a artículos relacionados. Esto conllevaría desarrollar una función de búsqueda de relacionados, que por ejemplo tuviese un fallback a artículos recientes en caso de no encontrarlos. La otra función que falta más notablemente es la inclusión de un usuario de tipo administrador. Los usuarios normales perderían la capacidad de actualizar sus publicaciones para que tuviesen el estado de publicadas, recayendo esta responsabilidad en el Admin. Con la gema “devise” podría implementarse un sistema de correos que avisara al administrador de cuando se emite una solicitud de publicación. Por último, una falta en la aplicación sería la capacidad de realizar múltiples subidas de imágenes en los artículos, pero esta falta se suple con la inclusión del formato MarkDown, con el que es posible incluir imágenes desde urls en internet embebidas en el texto.