UNIVERSIDAD DE BELGRANO

Las tesinas de Belgrano

Facultad de Ingeniería y Tecnología Informática
Carrera Ingeniería Informática

Desarrollo de un agregador RSS basado en web con filtros, notificaciones y capacidad de compartir fuentes

N° 727 Juan Wolfgang Alexander Becker

Tutor: Ing. Roberto Guglielmino

Departamento de Investigaciones
Fecha de defensa tesina 4 de mayo de 2015

Universidad de Belgrano
Zabala 1837 (C1426DQ6)
Ciudad Autónoma de Buenos Aires - Argentina
Tel.: 011-4788-5400 int. 2533
e-mail: invest@ub.edu.ar
url: http://www.ub.edu.ar/investigaciones

Índice

1. Resumen

1.1. Abstract

En la actualidad existe un sinfín de aplicaciones web. Muchas de estas se basan en contenido generado por usuarios. También existen muchas que obtienen el contenido de otras páginas y aplicaciones. En el primer caso se suele hablar de “redes sociales”, en el segundo de “mashups” (un término menos popular). Generalmente ambos conceptos no se contraponen sino que se complementan. Este último es el caso de la aplicación web de la que trata el presente trabajo. La aplicación desarrollada permite que sus usuarios agreguen fuentes de noticias (RSS/Atom) a sus cuentas, facilitando la lectura y el manejo de los contenidos provenientes de estas, e incluso logrando que diversos usuarios puedan compartir estos contenidos de forma simple.
El trabajo detalla las tecnologías utilizadas en el desarrollo de la aplicación mencionada, así como metodologías y herramientas de desarrollo que intervienen, y explica los pasos que se llevaron a cabo para llegar al producto final.

1.2. Justificación

Se decidió desarrollar una aplicación web y escribir acerca del proceso de desarrollo de la misma por diversas razones, mayormente de naturaleza subjetiva. Ante todo el desarrollo de aplicaciones web es un tema de interés personal que constituye el presente y promete ser el futuro del uso de aplicaciones.
La característica más motivadora de las aplicaciones de este tipo es la **ubicuidad**, en concreto la capacidad de acceder a ellas desde cualquier lugar (siempre cuando se tenga acceso a Internet). De esta forma una aplicación ubicada físicamente en un servidor de latinoamérica puede ser utilizada remotamente desde cualquier otro punto, no sólo a nivel mundial, sino también (con vista al futuro) a nivel universal.
Otro atractivo de las aplicaciones web es la **portabilidad**, que con el tiempo se parece estar perfeccionando cada vez más al tender a la estandarización progresiva de tecnologías web. Portabilidad es la capacidad de cierto software de poder ejecutarse en diferentes plataformas sin que esto signifique tener que generar código o soluciones particulares para algunas de estas. En concreto, las plataformas a las cuales se refiere en este contexto son los navegadores web, en los que se ejecuta parte de una aplicación. Los principales navegadores son IE, Chrome, Firefox, Safari y Opera, sin considerar las versiones de cada uno. El hecho de poder ejecutar una misma aplicación en cada uno de estos, significa un alto grado de portabilidad, consecuencia favorable de grandes esfuerzos de estandarización de las tecnologías subyacentes.

Se decidió desarrollar particularmente un agregador RSS/Atom por la necesidad propia de encontrar una manera rápida y eficiente de obtener noticias de interés de diversos sitios web.
El nivel de dificultad del desarrollo es acorde con el nivel de conocimientos y experiencia con el que se cuenta. Por lo tanto, el objetivo de desarrollar la presente aplicación resultó viable, una vez establecidos el alcance y las limitaciones que se listan a continuación.

1.3. Alcance

La aplicación **permite**:

- Crear, modificar y eliminar usuarios
- Agregar (‘seguir’)/quitar usuarios.
- Crear, modificar y eliminar categorías de fuentes.
- Agregar fuentes a las categorías.
- Asociar filtros a cada fuente agregada.
- Asociar notificaciones a cada fuente agregada.
- Visualizar las entradas actuales de las fuentes agregadas.
- Marcar entradas de feeds como favoritas.
- Filtrar la visualización de entradas por favoritos, leídos y no leídos.
- Guardar permanentemente entradas marcándolas como favoritas.

1.4. Limitaciones

Si bien durante este proyecto se generan archivos y directorios para realizar testing, estos no se van a utilizar por cuestiones de tiempo y restricciones en el tamaño del proyecto. Sin embargo se van a generar en forma implícita para facilitar pruebas futuras para potenciales proyectos que se basen en esta aplicación.

No permite:

- Almacenar feeds, excepto aquellos que estén marcados como favoritos.
- Comunicarse con otros usuarios.
- Aplicar filtros o notificaciones sobre categorías.
- Compartir filtros o notificaciones entre los usuarios.
- Exportaciones de ningún tipo.
- Ver entradas que por orden de publicación superen una cierta cantidad de entradas por feed.

En cuanto al desarrollo del proyecto, cabe aclarar que no se pretende redactar un paso a paso de cómo se implementa la aplicación. Se hará énfasis únicamente en las características y decisiones de mayor impacto, sin crear un historial completo de la aplicación. Es decir, no se podrá recrear la aplicación siguiendo simplemente los pasos descritos en las iteraciones.

Sí habrá referencias directas a parte del código para poder ver la implementación concreta de algunas funcionalidades. Además de adjuntarse el código fuente de esta aplicación, también se podrá acceder al mismo mediante el repositorio de github: https://github.com/donboli/socialfeed.git. De esta forma se podrá descargar el código entero con el historial correspondiente y tener una visión precisa de cómo se creó la aplicación.

2. Introducción

El trabajo consta del desarrollo de una aplicación web, así como de la documentación de dicho proceso. En concreto se trata de un agregador RSS (Atom) que sirve para acumular fuentes informativas, como por ejemplo de noticias, con el fin de poder centralizar la información, aplicar filtros y crear notificaciones. Además se podrá buscar otros usuarios de la aplicación y acceder a las fuentes de estos. Cada feed representa una fuente de información, que al ser consultada devuelve un conjunto de artículos recientes en forma de entradas (‘entries’).

La aplicación guarda los feeds y actualiza las entradas correspondientes dentro de un intervalo de tiempo que se repite constantemente. Esto sucede aún cuando no se está conectado, con el fin de poder disparar potenciales notificaciones establecidas por el usuario.

Si bien las entradas se guardan en la base de datos, estas se eliminan a partir de un máximo de entradas por feed. Las únicas entradas que permanecen indefinidamente en la base de datos, son aquellas que se marcan como favoritas. Además las entradas pueden estar marcadas como leídas o no leídas.

3. Marco Conceptual/Teórico

3.1. Antecedentes

Se eligió el framework **Ruby on Rails** (en adelante también RoR o Rails) para el presente proyecto, dado que utiliza Ruby como lenguaje de programación (un lenguaje muy fácil e intuitivo de entender), además de contar con una gran comunidad que sirve de apoyo, y haber sido utilizado por muchas aplicaciones web actuales. Entre los sitios basados en Rails se encuentran Twitter, Shopify, Crunchbase, Groupon, Bloomberg, Indiegogo, Airbnb, Slideshare y Soundcloud entre otros.

Como herramienta de organización y gestión del proyecto se utilizó Trello, una aplicación web gratuita que provee una interfaz visual para manejar tareas en forma de tarjetas. **Trello** permite organizar trabajos con Kanban, una metodología de desarrollo muy utilizada actualmente para proyectos de software. Sin embargo, el proyecto no intenta seguir con precisión la metodología anteriormente mencionada, para evitar posibles limitaciones ya sea de la herramienta (Trello) o del mismo proyecto. Trello es utilizado por muchas organizaciones de amplio renombre como Google, Adobe, The New York Times, Paypal, Spotify, entre otros.

Con el fin de versionar la aplicación web producto de este proyecto, se adoptó la herramienta **Git** y se creó un repositorio en Github. Git es un sistema distribuido de control de versionado, que permite persistir los avances en la implementación paso a paso. Además contiene muchas utilidades que facilitan la documentación y el manejo del código. Github es una aplicación web que ofrece mantener repositorios de Git en la web, con el fin de mantener una copia accesible de respaldo y promover colaboración con otras personas, entre otras características útiles.

Github es utilizado por empresas reconocidos como Paypal, SAP y Vimeo[3], mientras que Git en sí es utilizado además por Google, Linkedin, Microsoft, Facebook, Twitter, Netflix, etc.

Fuentes

- Siepen, D. (26 de Febrero de 2014). Top 15 sites built with Ruby on Rails. Consultado el 6 de Marzo de 2015, Obtenido de https://thecoderfactory.com/posts/top-15-sites-built-with-ruby-on-rails
- Trello. (n.d.). Consultado el 6 de Marzo de 2015, Obtenido de https://trello.com/home
- It keeps getting better. (n.d.). Consultado el 8 de Marzo de 2015, Obtenido de https://enterprise.github.com/
- Git. (n.d.). Consultado el 8 de Marzo de 2015, Obtenido de http://git-scm.com/

3.2. RSS/Atom

RSS y Atom son 2 de los formatos más utilizados para el envío periódico de contenidos web. Muchos sitios web adoptan una de estas tecnologías con el fin de proveer una manera alternativa de distribuir su contenido.
Los formatos de este tipo eliminan la necesidad de consultar cada una de las páginas de interés por separado. Además libera a los usuarios de tener que registrarse a la lista de envío de mails de las páginas, logrando que puedan mantener mayor privacidad.
Técnicamente tanto RSS como Atom están constituidos por archivos XML, llamados feeds o canales, que describen la fuente de la cual se consultan las noticias. A estas últimas se las refiere como “entradas”.
Con el fin de poder leer los feeds RSS/Atom se utiliza comúnmente un software conocido como lector o agregador de RSS. Los agregadores pueden formar parte de otras aplicaciones (como clientes de email) o constituir una aplicación por sí mismos. A su vez, es posible encontrar lectores para desktop, mobile o web[5].

Particularmente Atom tiene las siguientes ventajas frente a RSS[6]:

- Licencia menos restrictiva.
- Tipo de datos MIME registrado en IANA (Internet Assigned Numbers Authority).
- Espacio de nombre propio en XML (Extensible Markup Language).
- Soporte para URIs (Uniform Resource Identifier) relativas.
- Es un estándar de IETF (Internet Engineering Taks Force).

Fuentes

- What Is RSS? RSS Explained. (n.d.). Consultado el 9 de Marzo de 2015, Obtenido de http://www.whatisrss.com/
- RSS vs Atom . which one is better ? (5 de Agosto de 2012). Consultado el 9 de Marzo de 2015, Obtenido de https://shafiq2410.wordpress.com/2012/08/05/rss-vs-atom-which-one-is-better/

3.3. Metodologías Ágiles

Las metodologías ágiles en general proponen un flujo de trabajo que permite responder frente a la retroalimentación en base al desarrollo iterativo de un producto, reconociendo que los requisitos pueden cambiar constantemente. Estas metodologías brindan una alternativa a la dirección tradicional de proyectos.

A diferencia de metodologías secuenciales, que proponen un desarrollo por etapas, las metodologías ágiles promueven el desarrollo de funcionalidades priorizadas según su impacto o relevancia, para ser desarrolladas una por vez (en iteraciones). De esta manera es posible que las funcionalidades puedan ser aprobadas o rechazadas de inmediato por el interesado, permitiendo ajustar los requisitos y redirigir el equipo de desarrollo, en caso de ser necesario. En las iteraciones subsecuentes se puede tener en cuenta los cambios señalados por el interesado y mantener el producto alineado con las necesidades. Se asegura, por lo tanto, que al concluir el desarrollo se tendrá un producto final que responda plenamente a las necesidades del usuario, aún cuando estas últimas hayan cambiado por completo.
Con las metodologías tradicionales suele ocurrir que el producto final se vuelve en gran medida irrelevante debido al cambio de requisitos, al cual no se puede responder, dada la falta de retroalimentación por parte del interesado[7].

3.3.1. Scrum

Scrum es la metodología ágil más popular por su simplicidad y flexibilidad. Con ella se enfatiza la retroalimentación empírica, la autogestión de equipos, y el esfuerzo por crear incrementos de producto probados de forma apropiada dentro de iteraciones (sprints).
Los involucrados pueden asumen al menos uno de los siguientes roles: Propietario del producto, equipo, Maestro de Scrum. Entre estos se reparte la responsabilidad del rol tradicional del director de proyectos.
Además Scrum propone 5 reuniones: Preparación de Backlog, planificación de Sprint, Scrum diario, Reunión de revisión de Sprint, Reunión retrospectiva de Sprint[7]. Para aclarar, cada Sprint está conformado por unidades de trabajo conocidas como Scrums. Generalmente los Sprints tienden a ser semanales o de 2 semanas, mientras que el Scrum normalmente es diario.

En el presente proyecto no se utiliza Scrum principalmente por las siguientes razones:

- El proyecto consta de una única persona. Si bien esa persona podría cumplir con los 3 roles designados por Scrum, carecería de sentido considerando que los roles fueron concebidos para ser ocupados por personas o grupos de personas diferentes. Considerar una misma persona cumpliendo 3 roles diferentes solamente crearía complicaciones en este contexto.
- No se realizan las reuniones por la razón anterior.

No obstante se utilizan elementos propuestos por Scrum, como la retroalimentación empírica (en forma transparente), la autogestión, y la preparación y mantenimiento de un backlog de tareas.

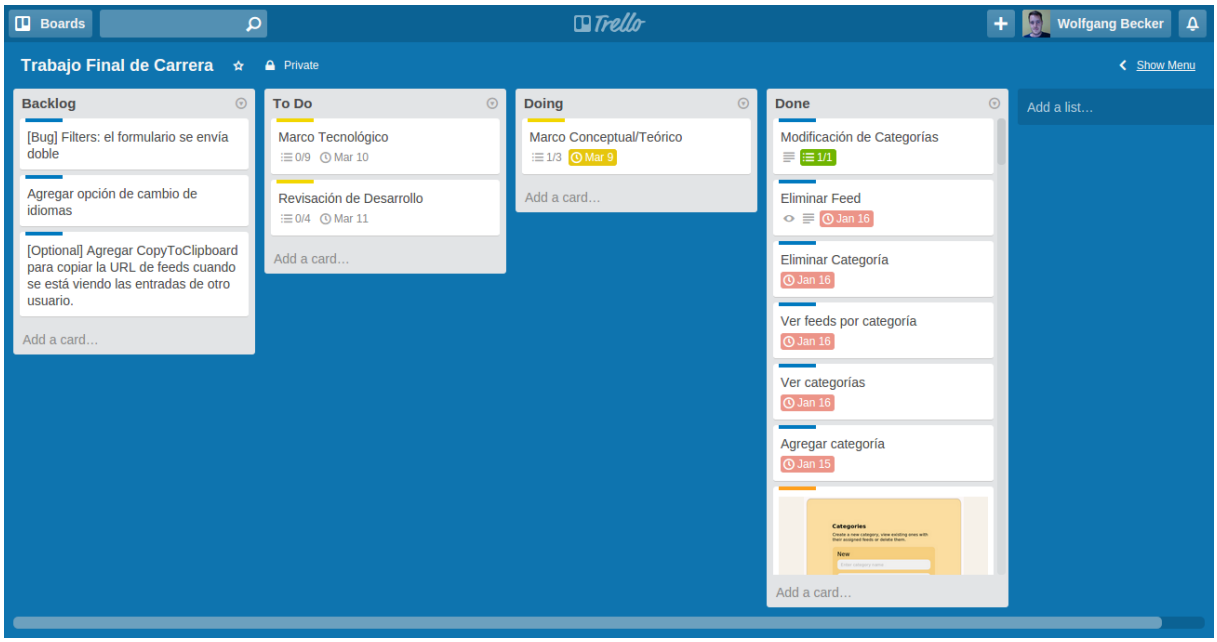
Fuente

- Agile Methodology. (n.d.). Consultado el 9 de Marzo de 2015, Obtenido de http://agilemethodology.org/

3.3.2. Kanban (Trello)

Kanban es una técnica de desarrollo de software con orígenes en el sistema de producción JIT (just-in-time) de Toyota. Aunque la producción de software se distingue de la producción masiva de líneas de montaje por el aspecto creativo de sus tareas, el mecanismo de gestión puede ser aplicado de todas formas.
La metodología Kanban consiste en manejar tareas o características simbolizadas por etiquetas, que se llevan a cabo al migrar desde un lado de un tablero al otro, pasando por un grupo de listas que simbolizan estados de las etiquetas. En principio las listas pueden ser rotuladas según las necesidades del proyecto particular, pero tradicionalmente los nombres son: “Pending”, “Analysis”, “Development”, “Test”, “Deploy”[8].

Trello es una aplicación web que sirve de herramienta para implementar un tablero de Kanban. De todas formas, esta es solo una de las posibilidades que se tiene al utilizar Trello, dado que las columnas se dejan definir libremente en cantidad, orden y nombres.
Con esta herramienta se realiza el seguimiento de todas las tareas del proyecto. En este caso creamos las columnas de "Backlog” (para tareas que se llevarán a cabo más adelante), “To Do” (tareas próximas a realizar), “Doing” (tareas actualmente en proceso) y “Done” (tareas finalizadas), para delimitar el avance de las tareas. Esto no refiere solamente al desarrollo sino también al análisis, diseño del producto y redacción del presente trabajo. Se optó por no utilizar los nombres de listas propuestos por Kanban, para poder englobar tareas y etapas que forman parte del proyecto en un marco más amplio y genérico que el mero desarrollo. En definitiva, el proyecto completo se planifica mediante Trello. A continuación se muestra una captura de pantalla con el estado actual del proyecto en esta herramienta.



Fuente

- What is Kanban? (n.d.). Consultado el 9 de Marzo de 2015, Obtenido de http://kanbanblog.com/explained/

3.3.3. Lean Software Development

Al igual que la metodología Kanban, Lean Software Development (en adelante LSD) adoptó sus principios del sistema de producción Just-In-Time de Toyota. Se realizaron adaptaciones al desarrollo de software, especialmente impulsados por los autores Mary y Tom Poppendieck, llegando a conformar los siguientes **7 principios**:

1. **Eliminar residuos**, estos son:
    a. Código o funcionalidad innecesarios.
    b. Comenzar más de lo que se pueda terminar.
    c. Retrasos en el proceso de desarrollo de software.
    d. Requerimientos poco claros o de cambio constante.
    e. Burocracia.
    f. Comunicación lenta o ineficiente.
    g. Trabajos parcialmente terminados.
    h. Problemas de defectos y calidad.
    i. Cambios de tareas.
2. **Incluir calidad** a través de:
    a. Programación en parejas.

   b. Test Driven Development: Metodología de desarrollo que impone la implementación de pruebas automatizadas, antes de escribir el código a probar.
   c. Retroalimentación constante.
   d. Minimización de tiempo entre etapas: Por ejemplo, corregir errores a medida que se reconocen, en vez de registrarlos y aplazar su tratamiento.
   e. Integración frecuente.
   f. Automatización.
   g. Manejo de compensaciones.
3. **Crear conocimiento** mediante:
   a. Programación en parejas.
   b. Revisiones de código.
   c. Documentación.
   d. Wiki.
   e. Código exhaustivamente comentado.
   f. Sesiones para compartir conocimientos.
   g. Entrenamiento.
4. **Aplazar compromisos** (especialmente cuando los requerimientos cambian rápidamente y las decisiones son irreversibles o costosas).
5. **Entregar rápidamente** por medio de:
   a. Personas que piensan por sí mismas, resuelven problemas, son proactivas y flexibles, toman acciones y decisiones apropiadas.
   b. Mantener las cosas simples.
   c. Trabajo en equipo.
   d. Eliminar residuos.
   e. Inclusión de calidad.
6. **Respetar las personas:**
   a. Respondiendo rápidamente.
   b. Escuchando atentamente.
   c. Escuchando opiniones sin descartarlas si son diferentes a las propias.
   d. Alentando la expresión de las personas.
   e. Mostrando empatía.
   f. Dando la responsabilidad para que las personas tomen decisiones.
   g. Siendo positivo y discrepando con un punto de vista, sin sonar agresivo, amenazante o discutidor.
7. **Optimizar el total**, lo cual se puede lograr, por ejemplo, organizando equipos por producto.

La presente Tesina no sigue estrictamente estos principios (debido principalmente a que el equipo consta de una única persona). Además no se tomaron medidas importantes de inclusión de calidad (como por ejemplo Test Driven Development), dada la falta de experiencia y conocimiento en este área, además de restricciones de tiempo. Los principios que sí se aplicaron al desarrollo son la eliminación de residuos, aplazamiento de compromisos y la entrega rápida.

Fuente

- Waters, K. (16 de Agosto de 2010). 7 Key Principles of Lean Software Development. Consultado el 24 de Marzo de 2015, Obtenido de http://www.allaboutagile.com/7-key-principles-of-lean-software-development-2/

3.3.4. Minimum Viable Product

Minimum Viable Product (en adelante MVP) es una técnica de desarrollo en la cual se crea un producto nuevo con características suficientes para satisfacer usuarios tempranos. Las características adicionales, que completan el producto, son diseñadas y desarrolladas recién luego de considerar la retroalimentación proveniente de los usuarios iniciales.
MVP propiamente dicho, constituye el producto en su estado inicial, creado con el propósito de ser completado por medio de los aportes de los usuarios.
Para ser exitoso un MVP debe cumplir con las siguientes 3 características:

- Cuenta con el valor necesario para que las personas estén dispuestas a utilizarlo o comprarlo.
- Demuestra suficiente beneficio a futuro para que los usuarios lo retengan.
- Provee un sistema de retroalimentación para guiar el desarrollo futuro.

Es importante que los usuarios iniciales puedan entender la visión o promesa que intenta cumplir el producto final, para que la retroalimentación guíe el producto en la dirección correcta.

La aplicación web desarrollada en esta Tesina puede constituir un MVP. Como tal, provee las funcionalidades mínimas necesarias para poder ser utilizada. Además, su diseño intenta promover la interacción entre usuarios, por medio de sus fuentes de información. Sin embargo, aún requiere un mecanismo de retroalimentación (por ejemplo una sección para feedback), para poder evolucionar en base a la demanda de los usuarios.

Fuente

- Janssen, C. (n.d.). What is a Minimum Viable Product (MVP)? - Definition. Consultado el 24 de Marzo de 2015, Obtenido de http://www.techopedia.com/definition/27809/minimum-viable-product-mvp

4. Marco tecnológico

4.1. Ruby

Ruby es un lenguaje de programación dinámico, de código abierto con un enfoque en simplicidad y productividad. Tiene una sintaxis elegante, natural de leer y fácil de escribir.
Se publicó en 1995 y en 2006 obtuvo aceptación masiva por la comunidad de programadores. Generalmente está ubicado entre los 10 lenguajes de mayor crecimiento y popularidad según numerosos índices (como el índice TIOBE por ejemplo). Gran parte de su crecimiento es debido a la popularidad del software implementado con Ruby, como es el caso del framework web Ruby on Rails.
Además es completamente gratuito, con la libertad de ser utilizado, copiado, modificado y distribuido.
Las características más importantes de este lenguaje son:

- Todo es un objeto.
- Es flexible, dado que permite modificar libremente sus partes.
- Permite herencia simple, con la posibilidad adicional de implementar mixins con módulos.
- Uso de closures (bloques): Permite definir el comportamiento interno de un método, pasando un bloque de código.

Fuentes

- Ruby. (n.d.). Consultado el 9 de Marzo de 2015, Obtenido de https://www.ruby-lang.org/en/
- Ruby. (n.d.). Consultado el 9 de Marzo de 2015, Obtenido de https://www.ruby-lang.org/en/about/

4.1.1. Gemas

Al igual que la mayoría de los lenguajes de programación, Ruby está asociado a un amplio espectro de librerías creadas por terceros. La mayoría de estas se publican como *gemas*. Las gemas se pueden crear, compartir e instalar mediante un sistema de empaquetado conocido como RubyGems.
El principal repositorio de gemas es RubyGems.org. Las gemas correspondientes pueden ser buscadas y descargadas directamente de este sitio o por medio del comando gem. En la actualidad las gemas se encuentran generalmente publicadas en RubyGems.org, mientras que el repositorio de las mismas se ubica en Github (ver 4.5.1.1).

Para instalar una gema se cuenta generalmente con indicaciones en la página del repositorio (readme.md). Sin embargo, los pasos generales tienden a ser los siguientes:

1. Agregar la gema al Gemfile. **gem <nombre de gema>**
2. Correr **bundle install** en la terminal.
3. Si la gema incluye JavaScript y/o CSS normalmente se agrega la dependencia al archivo manifest correspondiente. En el caso de JavaScript esto se realiza agregando ***//= require jquery*** al archivo application.js. Para CSS se debe agregar ****= require_self*** al archivo application.css.
4. En caso de que el servidor esté corriendo, se recomienda reiniciarlo antes de continuar con el desarrollo, con el fin de asegurar que se adopten los cambios.

En este espacio se explica brevemente las funcionalidades y características que aportan algunas de las gemas más importantes utilizadas en este proyecto. Con el fin de limitar la longitud de esta sección se han excluido aquellas gemas que no aportan funcionalidad a la aplicación en un entorno de producción. Esto último incluye aquellas gemas destinadas a probar la aplicación (ejemplo: Factory Girl y RSpec), así como herramientas que brindan apoyo al desarrollo (ejemplo: Erb2Haml e i18n-tasks). Además se excluyen gemas que se utilizan para agregar plugins de JavaScript, la mayoría de las gemas que forman parte de Rails por defecto, y otras por no tener relevancia mayor en este proyecto, según criterio subjetivo.

Fuente

- Ruby. (n.d.). Consultado el 9 de Marzo de 2015, Obtenido de https://www.ruby-lang.org/en/libraries/

4.1.1.1. I18n

Esta gema, que se encuentra integrada por defecto en Ruby on Rails (a partir de la versión de Rails 2.2), provee un conjunto de herramientas para facilitar la traducción de una aplicación a uno o múltiples idiomas. Además de traducción también se cuenta con *localización*, lo cual significa que según la región o país se puede cambiar características como sistemas de medición o tipos de moneda.
Principalmente se recurre a abstraer o reemplazar todas las cadenas de caracteres destinadas a ser visualizadas, con el fin de procesar estas antes de que las páginas correspondientes sean renderizadas. De esta forma, dependiendo de la configuración, se puede mostrar la aplicación en el idioma adecuado.
En el caso particular de este proyecto se utiliza por defecto el idioma español, aunque también se puede cambiar a inglés.

El nombre I18n se debe a que es un acortamiento de la palabra “Internationalization”, siendo 18 letras entre la “I” inicial y la “n” final.
Algunas características de I18n son:

- Traducción y localización
- Interpolación de valores a traducciones
- Pluralización
- Transliteración personalizable a ASCII
- Valores y configuraciones por defecto flexibles
- Búsqueda de traducciones por mayor

Fuentes

- Fuchs, S., Harvey, J., Soller, S., Moore, S., & Aimonetti, M. (2008). Svenfuchs/i18n. Consultado el 9 de Marzo de 2015, Obtenido de https://github.com/svenfuchs/i18n
- Fuchs, S., & Minařík, K. (n.d.). Guides.rubyonrails.org. Consultado el 10 de Marzo de 2015, Obtenido de http://guides.rubyonrails.org/i18n.html

4.1.1.2. Devise

Devise es una gema flexible que facilita la autenticación para aplicaciones creadas en Rails. Es una solución que respeta el modelo MVC de Rails, permite tener múltiples modelos conectados al mismo tiempo, y está compuesta por un conjunto de módulos independientes que permiten adaptar la gema a las necesidades particulares de la aplicación.

Los 10 módulos que conforman Devise son:

- Database Authenticable: Encripta y almacena una clave en la base de datos para validar la autenticidad de un usuario durante su conexión.
- Omniauthable: Agrega soporte para OmniAuth (una gema que permite autenticación por proveedores externos).
- Confirmable: Envía emails con instrucciones de confirmación y verifica si una cuenta ya ha sido confirmada antes de abrir una sesión.
- Recoverable: Restablece la contraseña de un usuario y envía instrucciones para generar una nueva.
- Registerable: Maneja el registro de usuarios por medio de un proceso, permitiéndoles editar y destruir su cuenta.
- Rememberable: Maneja la generación y limpieza de *tokens* para recordar al usuario desde un cookie guardado en el navegador.

- Trackable: Mantiene la cuenta de la cantidad de veces que se conectó un usuario, además de marcas de tiempo y direcciones IP.
- Timeoutable: Expira sesiones que no han mostrado actividad por un cierto período de tiempo.
- Validatable: Provee validaciones de email y contraseña. Puede ser personalizado para las validaciones necesarias.
- Lockable: Bloquea una cuenta luego de cierta cantidad de intentos fallidos de conectarse. Se puede desbloquear mediante email o luego de un cierto período de tiempo.

Para la aplicación del presente proyecto se utilizaron los módulos: Database Authenticable, Registerable, Recoverable, Rememberable, Trackable y Validatable.

Fuente

- Plataformatec/devise. (2009). Consultado el 9 de Marzo de 2015, Obtenido de https://github.com/plataformatec/devise

4.1.1.3. Feedjira

Feedjira sirve para manejar los feeds RSS/Atom así como sus entradas. Es una pieza fundamental en el desarrollo de la presente aplicación.
La gema está diseñada para obtener y analizar feeds con énfasis en velocidad. Una vez presentes los feeds, estos se pueden actualizar utilizando los objetos de feed obtenidos. Feedjira se ocupa de insertar automáticamente metadatos, proveniente de las respuestas de headers HTTP, para aminorar el uso del ancho de banda, eliminar análisis innecesarios, y lograr rapidez general.
La ventaja de Feedjira frente a otras herramientas, no es sólo la velocidad sino también el soporte para distintos tipos y versiones de formatos de sindicación, siendo los principales RSS y Atom. Esto no es siempre el caso con las herramientas que compiten con Feejira.

Fuentes

- Allured, J., Dix, P., Kirch, J., & Templin, E. (n.d.). Feedjira/feedjira. Consultado el 9 de Marzo de 2015, Obtenido de https://github.com/feedjira/feedjira
- Feedjira. (n.d.). Consultado el 9 de Marzo de 2015, Obtenido de http://feedjira.com/

4.1.1.4. Sidekiq/Sidetiq

Sidekiq es un framework para procesamiento de tareas en background. Entre otras funcionalidades, permite aminorar tiempos de respuesta liberando el proceso principal de una aplicación, de manera que una tarea costosa se lleve a cabo en un proceso separado. En el caso de la presente aplicación se logra actualizar la base de datos sin afectar el uso de la aplicación por parte de los usuarios. Si no se utilizara procesamiento en background para estas tareas, y en cambio el mismo proceso de servidor web se encargara de realizarlas, significaría que los usuarios tendrían que esperar para poder seguir usando la aplicación, cada vez que se actualizara la base de datos.

La activación de tareas en Sidekiq es manual. Es decir, desencadenar la ejecución depende de un agente externo. Normalmente el enfoque es activar los procesos desde la aplicación web cuando se cumplen determinadas condiciones. En el caso del proyecto es necesario actualizar la base de datos periódicamente, con lo cual conviene automatizar las tareas en background para lograr que sean independientes de la aplicación.
Con este propósito se creó otra gema llamada Sidetiq, que utiliza Sidekiq para realizar tareas recurrentes. Sidetiq permite planificar la ejecución de tareas por medio de un conjunto de métodos que especifican la periodicidad de ejecución. Esto puede ser por segundos, minutos, horas, dias, semanas, meses, así como horarios o meses específicos por año, entre otras opciones.
En el proyecto se utiliza estas gemas en conjunto en 2 ocasiones:

1. Actualizar las entradas de los feeds cada 15 minutos.
2. Limitar la cantidad de entradas por feed una vez cada 24 horas, eliminando todas las entradas que no están entre las 50 más recientes de cada feed.

Fuentes

- Simple, efficient message processing for Ruby. (n.d.). Consultado el 9 de Marzo de 2015, Obtenido de http://sidekiq.org/
- Perham, M. (n.d.). Mperham/sidekiq. Consultado el 9 de Marzo de 2015, Obtenido de https://github.com/mperham/sidekiq
- Svensson, T. (n.d.). Tobiassvn/sidetiq. Consultado el 9 de Marzo de 2015, Obtenido de https://github.com/tobiassvn/sidetiq

4.1.1.5. HoboFields

HoboFields forma parte de un plugin mayor para Rails (Hobo) que provee una serie de extensiones concebidas con el propósito de facilitar el desarrollo de aplicaciones web.

En el caso de este proyecto sólo se utiliza la extensión particular HoboFields, la cual facilita en gran medida el manejo de la estructura de la base de datos en relación a los modelos.
Para adaptar la base de datos subyacente a los modelos de Rails, por defecto, se tendría que escribir ***migraciones***. Esto son archivos ubicados en un directorio especial, que definen los campos y cambios de la base de datos escritos en Ruby de manera declarativa a través de un DSL (Domain Specific Language). Esto significa que las migraciones cuentan con un conjunto de métodos que se comunican con la base de datos subyacente y se traducen en las sentencias de SQL específicas que requiere la base de datos para cambiar su estructura. Con esto se ahorra tener que escribir SQL nativo, el cual puede variar según el gestor de base de datos que se esté utilizando. Además se evita dependencia del gestor de base de datos, dado que la definición de la base de datos forma parte de la aplicación y puede ser traducida automáticamente en el SQL particular que requiera el gestor.
Las migraciones son una buena respuesta a los problemas anteriores, pero se deben definir manualmente. Además los modelos no contienen una definición de los campos en la base de datos. Esto significa que se debe revisar las migraciones o el esquema de la base de datos para saber con qué campos cuenta cada modelo.
HoboFields soluciona la inconsistencia anterior, permitiendo que se definan los campos de la base de datos directamente en los modelos representativos. En base a la definición de los campos del modelo, HoboFields es capaz de crear automáticamente las migraciones necesarias en forma interactiva. Con esto se ahorra escribir las migraciones manualmente.
Esta gema se utiliza para definir los campos de todos los modelos del presente proyecto.

Fuente

- HoboFields. (n.d.). Consultado el 9 de Marzo de 2015, Obtenido de http://www.hobocentral.net/manual/hobo_fields

4.1.1.6. Simple Form

Simple Form es una solución flexible que facilita la definición de los formularios en las vistas de Rails. La meta principal de esta gema es evitar afectar la definición del layout, dejando el diseño a cargo del desarrollador. Gran parte de la nomenclatura se heredó de otra gema conocida como Formtastic, facilitando así su aprendizaje. Simple Form no provee una funcionalidad independiente de los métodos para crear formularios que vienen con Rails por defecto, sino que actúa mayormente de intermediario para estos, facilitando su uso.

Fuente

- Valim, J., Da Silva, C., França, R., & Ermolovich, V. (n.d.). Plataformatec/simple_form. Consultado el 9 de Marzo de 2015, Obtenido de https://github.com/plataformatec/simple_form

4.1.1.7. Draper

Desde un punto de vista de diseño de la aplicación, Draper es una gema que implementa un patrón conocido como *Decorador*.
Esta gema agrega una capa adicional a la lógica de presentación de la aplicación Rails. Permite separar la lógica de negocio de un modelo de su lógica de presentación. El modelo alberga solamente la lógica de negocio. Los aspectos de presentación se implementan en una entidad aparte conocida como *decorador*.
Normalmente si no se utiliza Draper, se tiende a sobrecargar los modelos o crear una serie de métodos *helpers* que no se encuentran ligados lógicamente al modelo al que responden. Con Draper se logra mantener un modelo liviano, mientras que se mantiene los métodos de presentación en una entidad separada pero asociada al modelo.
Para aplicar un decorador sobre el modelo correspondiente se debe llamar simplemente un método (decorate) sobre el modelo, el cual devuelve el mismo modelo encapsulado en un decorador. Dependiendo de la implementación, los métodos que se llaman sobre el decorador son tratados directamente por éste o delegados al modelo subyacente.

Fuente

- Casimir, J., Klabnik, S., Ermolovich, V., & Haines, A. (n.d.). Drapergem/draper. Consultado el 9 de Marzo de 2015, Obtenido de https://github.com/drapergem/draper

4.1.1.8. Acts As Tenant

En la presente aplicación se necesita manejar los datos de muchos usuarios que comparten una misma base de datos. Con el fin de separar los datos de cada uno, durante cada operación sobre la base de datos se realiza un filtrado oculto según el usuario conectado.
Por ejemplo, cuando un usuario está conectado y quiere obtener todos los feeds que tiene guardados en su cuenta, internamente se realiza un filtrado de los feeds según el ID del usuario, para evitar que se obtengan los feeds de los demás usuarios.
La necesidad de que esto se realice en forma automática, tiene 2 razones:

1. Seguridad: El desarrollador no debe olvidarse de separar los datos en ningún momento.
2. DRY (Don't Repeat Yourself): Evitar repetir el mismo código.

A su vez, Acts As Tenant permite cambiar el usuario o realizar operaciones sin el prefiltrado si fuera necesario. La única ocasión en la que se cambia de usuario en la presente aplicación, es cuando se accede a la vista de los feeds de dicho usuario.
Además, la gema viene preparada con un método que permite enfocar validaciones de unicidad a nivel de usuario. De manera que, por ejemplo, no se evita que 2 usuarios tengan el mismo feed agregado, pero sí que se agregue un feed por duplicado al mismo usuario.

Fuente

- Matthijssen, E. (2011). ErwinM/acts_as_tenant. Consultado el 9 de Marzo de 2015, Obtenido de https://github.com/ErwinM/acts_as_tenant

4.1.1.9. Ransack

Ransack permite crear formularios de búsqueda de diferentes niveles de complejidad. En el caso de la aplicación que se desarrolla en este proyecto se utiliza Ransack en 2 ocasiones:

1. Filtrado las entradas en la vista de feeds.
2. Filtrado de usuarios en la vista de Seguir Usuarios.

La gema debe ser implementada a nivel de formulario (en las vistas) y provee métodos que se deben utilizar en el controlador para recuperar la entidades buscadas. En el formulario se define el criterio de búsqueda de cada campo. Esto puede ser por texto completo, una parte, la parte inicial, la parte final, entre otras opciones. Otro característica es que se puede buscar por diversos atributos a la vez (como nombre y email) a través de un único campo.
Además de lo anterior, Ransack también permite crear links para ordenar el contenido encontrado. Sin embargo no se ha utilizado esta facilidad en la presente aplicación.

Fuente

- Miller, E. (1 de Enero de 2011). Activerecord-hackery/ransack. Consultado el 9 de Marzo de 2015, Obtenido de https://github.com/activerecord-hackery/ransack

4.1.2. RVM

Como la gran mayoría de los lenguajes de programación, Ruby viene en diversas versiones. En ocasiones es necesario instalar más de una versión, por razones de compatibilidad con ciertas gemas o plataformas.
Para el proyecto se utilizó la versión de Ruby 2.1.3. Para poder instalar el intérprete de esta versión existen diversas maneras de hacerlo. Generalmente la menos recomendada es la directa, dada la dificultad (instalar desde código fuente) y posterior falta de flexibilidad (reinstalar cada vez

que se quiere cambiar de versión). Por lo tanto se decidió utilizar RVM para instalar la versión de Ruby de este proyecto. Con esta herramienta se puede tener múltiples versiones instaladas al mismo tiempo y seleccionar una por vez para ser utilizada.
Además RVM permite crear *gemsets*, que son grupos de gemas de diferentes versiones. Estos grupos se pueden adoptar para realizar pruebas de compatibilidad con Ruby.

Fuente

- Seguin, W. (n.d.). ∞The Basics of RVM. Consultado el 12 de Marzo de 2015, Obtenido de https://rvm.io/rvm/basics

4.1.3. Rails

Rails es un framework de desarrollo de aplicaciones web escrito en el lenguaje Ruby. Un framework es en este caso la implementación de una arquitectura como base de desarrollo. Está diseñado para facilitar el desarrollo de aplicaciones web, mediante suposiciones sobre lo que necesita cada desarrollador para poder comenzar. Permite un alto nivel de productividad que se traduce en escribir menos código, mientras se logra más que en muchos otros lenguajes y frameworks.
Al hacer suposiciones tanto estructurales como de configuración, Rails impone en gran parte una metodología de desarrollo. Por lo tanto al desarrollar con este framework posiblemente se tenga que abandonar algunas prácticas provenientes de otros ámbitos o metodologías. En caso contrario, es probable que se tendrá problemas.
Rails se instala como una gema de Ruby. Sin embargo no se incluyó en la sección de gemas, para destacar su importancia en el presente proyecto.

El framework incluye 2 principios de orientación:

- DRY (Don't Repeat Yourself): Al no repetirse varias veces la misma información, el código se vuelve más mantenible, extensible y contendrá menos errores.
- Convención sobre Configración: Rails se basa en opiniones acerca de la mejor manera de resolver ciertos aspectos del desarrollo web. Esto permite implementar una aplicación sin previamente tener que definir la configuración.

Un aspecto importante de Rails es su diseño, basado en MVC (Model View Controller). Las características de este patrón se incluyen a continuación (ver 4.1.3.1). Además se destaca el mecanismo que utiliza Rails para persistir sus objetos en la base de datos de la aplicación (ver 4.1.3.2).

Fuente

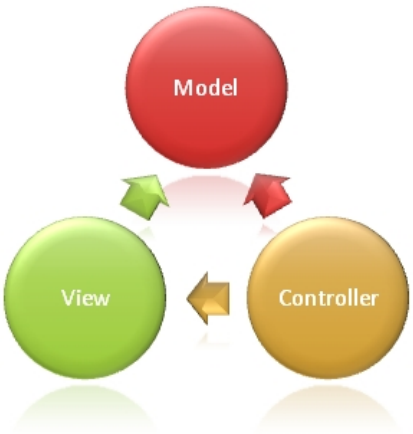
- Heinemeier Hansson, D. (n.d.). Guides.rubyonrails.org. Consultado el 13 de Marzo de 2015, Obtenido de http://guides.rubyonrails.org/getting_started.html

4.1.3.1. MVC

MVC (Model View Controller) es el nombre de un patrón de diseño, para crear aplicaciones web, formado por 3 componentes:

- Modelo: Representa el núcleo de la aplicación, también referida como la lógica de negocio (por ejemplo los usuarios o los feeds en la aplicación de este proyecto).
- Vista: Visualiza los datos por medio de templates.
- Controlador: Maneja las peticiones entrantes y actúa de intermediario entre las vistas y los modelos para crear una respuesta.

MVC provee además el control sobre HTML, CSS y JavaScript.
Esquemáticamente este patrón puede ser representado como sigue.



Model: Lógica de negocio

View: Visualización

Controller: Control de peticiones

El **modelo** maneja la lógica de los datos de la aplicación. Generalmente se ocupa de realizar la consulta, modificación y creación de datos en una base de datos (persistencia). En tal caso, esto puede ser realizado en forma directa, por ejemplo mediante SQL, o utilizando un sistema de ORM (Object-Relational Mapping) que realiza las operaciones en la base de datos de forma transparente.
La vista visualiza los datos provenientes del modelo. Consiste generalmente en HTML y/o algún sistema de plantillas como ERb o Haml. Estos últimos permiten preprocesar las vistas, con el fin de insertar datos en la página o alterar la visualización del marcado.

El **controlador** maneja la interacción con el usuario respondiendo a una petición entrante, proveniente de un cliente (ejemplo: navegador web). Dependiendo del tipo de petición entrante (Verbos del protocolo HTTP) se realizan acciones diferentes en el controlador.

La separación de responsabilidades en los 3 componentes de MVC simplifica el manejo de aplicaciones complejas, dado que permite el enfoque en diferentes aspectos en forma aislada. Por ejemplo se puede definir una vista sin depender de la lógica de negocio subyacente. Además facilita la realización de pruebas sobre la aplicación.

MVC también simplifica el desarrollo en equipos. Los desarrolladores pueden trabajar en paralelo en diferentes vistas, lógicas de controladores, y lógicas de negocio, de una misma aplicación.

Fuente

- ASP.NET MVC Tutorial. (n.d.). Consultado el 13 de Marzo de 2015, Obtenido de http://www.w3schools.com/aspnet/mvc_intro.asp

4.1.3.2. ActiveRecord

Para explicar lo que es ActiveRecord y qué problemas soluciona se debe entender el concepto más global de ORM primero.
ORM (Object-Relational Mapping): Es una técnica que conecta los objetos o estructuras de datos de una aplicación con las tablas correspondientes en un sistema gestor de base de datos relacional. Utilizando esta técnica se pueden almacenar y obtener fácilmente las propiedades y relaciones de los objetos de la aplicación desde la base de datos, sin tener que escribir SQL en forma directa y con menos código de acceso a la base de datos en general.

ActiveRecord es una implementación de un **patrón** homónimo que propone lo siguiente: Objetos consisten tanto en datos persistentes como en operaciones sobre dichos datos. ActiveRecord intenta educar a los usuarios de los objetos sobre las operaciones de escritura y lectura que se pueden realizar sobre sus datos. Esto lo realiza incluyendo la lógica de acceso a la base de datos en los objetos.

Los modelos de Rails están implementados con ActiveRecord, de manera que estos sirven de interfaz con las respectivas tablas de la base de datos. Se facilitan así la creación y el uso de objetos de negocio que requieren ser persistidos.
Los mecanismos que provee ActiveRecord son:

- Representar modelos y sus datos.

- Representar asociaciones entre dichos modelos.
- Representar jerarquías de herencia a través de modelos relacionados.
- Validar modelos antes de persistirlos en la base de datos.
- Realizar operaciones sobre la base de datos en forma orientada a objetos.

Fuente

- Heinemeier Hansson, D. (n.d.). Guides.rubyonrails.org. Consultado el 13 de Marzo de 2015, Obtenido de http://guides.rubyonrails.org/active_record_basics.html

4.2. HTML

HTML (HyperText Markup Language) es un lenguaje utilizado para describir el contenido de una página web. Consiste en una sintaxis que permite crear elementos que encapsulan contenido dentro del documento, para que un *agente de usuario* (un navegador web por ejemplo) pueda interpretar su significado.
Este lenguaje es una representación en texto plano del contenido y su significado. HTML no se debe confundir con un lenguaje de programación. La intención no es operar sobre contenido, sino darle un significado.
Un *agente de usuario* es cualquier software que se utilice para acceder a una página web, por parte de un usuario. Los navegadores web tanto desktop como móviles forman parte de este grupo, pero no son los únicos agentes de este grupo. A este grupo también pertenecen, por ejemplo, los programas que trabajan sobre las páginas con el fin de optimizar búsquedas, aunque no haya un usuario que los controle en forma directa.
En el caso de la aplicación web, que se desarrolla como parte de este proyecto, no se utiliza HTML en forma directa. Sin embargo, se utiliza un sistema de plantillas llamado Haml, el cual se preprocesa antes de ser enviado al agente de usuario, obteniendo HTML.

Fuente

- The basics of html. (n.d.). Consultado el 13 de Marzo de 2015, Obtenido de https://docs.webplatform.org/wiki/guides/the_basics_of_html

4.2.1. Haml

Haml (HTML Abstraction Markup Language) es un sistema de plantillas que simplifica el uso de HTML con Ruby embebido. Permite crear templates de HTML haciendo fuerte uso de indentaciones e incorpora código Ruby con una sintáxis muy simple y acotada. En el proyecto

actual se optó por utilizar Haml con el fin de mejorar la legibilidad, aumentar la productividad y aminorar la cantidad de caracteres requeridos por template.
Al utilizar Haml se elimina las etiquetas de cierre de los elementos de HTML, con lo cual se ahorra escribir mucho código, se vuelve más legible y se evita la repetición. Esto es posible debido a que se utiliza la indentación para establecer la jerarquía de los elementos. Por un lado, se puede argumentar que esto genera dependencia de la indentación, pero por el otro, la indentación constituye una buena práctica en HTML, con lo cual forzarla es aún más favorable.
Se debe tener en cuenta que HAML siempre se preprocesa y traduce a HTML antes de enviarse al cliente. De esta manera el HTML generado puede ser minimizado para aminorar su tamaño, de forma que no se pierde la ventaja de HTML de poder eliminar espacios e indentaciones para aumentar la velocidad de envío.
En Rails la alternativa a Haml es ERB, que se utiliza por defecto. ERB resulta más fácil de utilizar cuando se acostumbra utilizar HTML normalmente. De hecho ERB utiliza directamente la sintáxis de HTML y agrega la sintáxis '<% %>' para incluir código Ruby en los templates. La desventaja de ERB es la mayor cantidad de caracteres requeridos, en muchos casos se requiere más del doble de caracteres en ERB que para un template equivalente en Haml.

Fuente

- Catlin, H., & Weizenbaum, N. (n.d.). About. Consultado el 13 de Marzo de 2015, Obtenido de http://haml.info/about.html

4.3. JavaScript

JavaScript (también referenciado como JS) es un lenguaje interpretado orientado a objetos, utilizado en millones de páginas y aplicaciones web en todo el mundo.
Es especialmente dinámico dado que está basado en el prototipado de objetos, que permite agregar atributos y funciones a los objetos en tiempo de ejecución. La sintaxis es intencionalmente similar a la de lenguajes populares como Java o C++ para facilitar su aprendizaje. Semánticamente las diferencias son mayores.
JavasScript puede utilizarse como un lenguaje procedural y/o orientado a objetos. Mediante el prototipado se puede crear objetos similares a partir de objetos preexistentes. Esto permite un funcionamiento comparable con la herencia en otros lenguajes, aunque el concepto no es intercambiable.
Si bien JavaScript también se utiliza para desarrollar aplicaciones del lado del servidor (como con node.js), una característica importante a tener en cuenta es que la ejecución se realiza normalmente en el cliente (generalmente un navegador web). Esto agrega complejidad a las implementaciones, dado que según el cliente se tiene diferentes intérpretes del lenguaje cuyos

resultados no siempre son los mismos. Por ejemplo: En el caso de Google Chrome se tiene el intérprete V8, mientras que en Mozilla Firefox se usa normalmente SpiderMonkey. El código que funciona en Chrome, no funciona necesariamente en Firefox y viceversa, dada las diferentes implementaciones de sus intérpretes. Más allá, los problemas pueden radicar en una brecha de eficiencia entre las interpretaciones de un mismo código.
A raíz de los problemas anteriores, surgieron varias librerías de JavaScript con el propósito de manejar las diferencias entre los navegadores, además de facilitar y agilizar el desarrollo en general. Algunos de los más populares son JQuery y Prototype.
En el desarrollo del presente proyecto se utilizó JQuery, por lo cual se profundizan las características del mismo en el próximo inciso (ver 4.3.1).

Fuentes

- About JavaScript. (n.d.). Consultado el 13 de Marzo de 2015, Obtenido de https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript
- JavaScript Libraries. (n.d.). Consultado el 13 de Marzo de 2015, Obtenido de http://www.w3schools.com/js/js_libraries.asp

4.3.1. JQuery

JQuery es una librería de JavaScript muy utilizada en el desarrollo de aplicaciones web. Provee funcionalidades diversas que solucionan problemas genéricos que se presentan muy frecuentemente, como por ejemplo la selección y manipulación de elementos HTML. La librería es pequeña con lo cual no afecta considerablemente el tiempo de carga de las páginas. Además es rápida, de manera que no desperdicia innecesariamente recursos del sistema en que se ejecuta.
Otra característica importante de JQuery es el hecho de estar optimizado para correr en los navegadores más populares (Cross-Browser), asegurando que las instrucciones de JavaScript se ejecutarán en cualquiera de estos. Puede parecer trivial, pero muchas veces las implementaciones en JavaScript de una misma funcionalidad para diversos navegadores puede ser sutilmente diferente. Con el fin de evitar escribir código específico para algunos navegadores, JQuery se ocupa internamente de resolver dichas inconsistencias.
Rails viene con JQuery cargado por defecto, con lo cual no es necesario llevar a cabo instalaciones particulares.

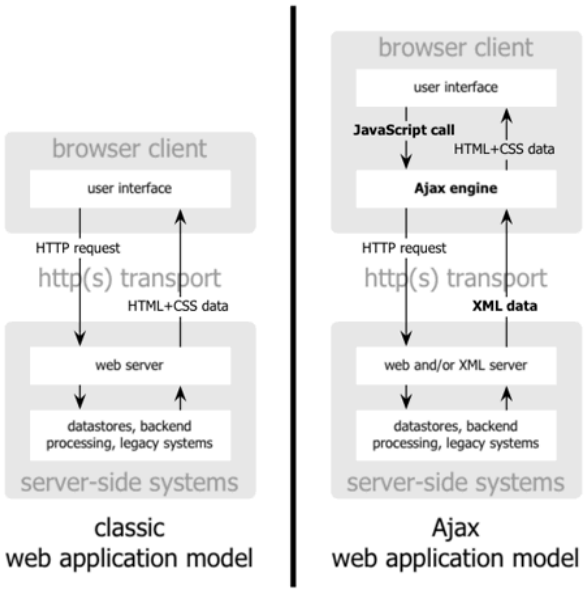
Fuente

- JQuery. (n.d.). Consultado el 13 de Marzo de 2015, Obtenido de http://jquery.com/

4.3.2. AJAX

AJAX (Asynchronous JavaScript And XML) es una manera de utilizar diversas tecnologías preexistentes con el fin de enviar datos entre el cliente y el servidor de una manera diferente a la tradicional. Normalmente el envío de datos desde una página web implica la recarga completa de la misma. Con AJAX se pueden enviar y recibir datos de manera que sólo se actualice una parte de la página. La respuesta del servidor no tiene que ser necesariamente un archivo HTML, sino que puede ser una estructura de datos como XML o JSON (4.4.3), con el fin de sólo recibir información, sin el código para el marcado y estilos correspondientes. Para insertar la información en la página generalmente se utiliza JQuery (4.4.1).

A continuación se incluye un esquema (Eguiluz J, 2015) para ilustrar la diferencia anteriormente mencionada entre al enfoque tradicional (de cargar la página completa), en contraposición a reemplazar una parte de dicha página (utilizando AJAX).



Fuente

- Eguiluz, J. (n.d.). Capítulo 1. Introducción a AJAX (Introducción a AJAX). Consultado el 15 de Marzo de 2015, Obtenido de http://librosweb.es/libro/ajax/capitulo_1.html

4.3.3. JSON

JSON (JavaScript Object Notation) es un formato liviano de intercambio de datos. Además de ser fácil de leer por humanos, también es simple para ser analizado y generado por máquinas. El formato está basado en un subconjunto del lenguaje JavaScript, al cual le debe parte de su

nombre. Aún así el formato es independiente de dicho lenguaje y puede utilizarse en cualquier lenguaje de programación.
El formato JSON se utiliza no sólo para comunicar el cliente con el servidor dentro de una misma aplicación, sino que también sirve para la comunicación entre aplicaciones, como en el caso de utilizar una API.
Otro formato muy conocido, que está siendo utilizado menos debido a la llegada de JSON, es XML. Dicho formato es un poco más complicado y su sintaxis es muy parecida a la de HTML.

Fuente

- Introducing JSON. (n.d.). Consultado el 15 de Marzo de 2015, Obtenido de http://json.org/

4.4. VCS (Version Control System)

Un VCS (Version Control System) es un sistema que permite grabar cambios realizados a un archivo o conjunto de archivos a lo largo del tiempo, de manera que se pueda volver a versiones anteriores en caso de ser requerido. Si bien puede ser utilizado para cualquier conjunto de archivos, generalmente se suele referir a proyectos de software.
Un sistema de este tipo permite revertir cambios en archivos particulares, revertir un proyecto entero a un estado anterior, comparar cambios a través del tiempo, ver quién realizó la última modificación que haya causado algún problema, ver quién insertó algún asunto para ser arreglado y en qué momento, entre otras funcionalidades.

Principalmente existen 3 tipos de VCS:

- **Locales**: Estos sistemas pueden ser sumamente simples. Por ejemplo, muchas personas crean una carpeta por cada versión de un proyecto. Al manejar los cambios de forma manual, es altamente probable cometer errores humanos. Por ejemplo se puede olvidar con qué carpeta se está trabajando actualmente o copiar archivos equivocados. También existen soluciones de este tipo más formales y técnicamente elaborados, como es el caso del sistema RCS (el cual utiliza una base de datos que guarda los cambios aplicados a los archivos). En este caso se eliminan gran parte de los potenciales errores causados por el manejo manual.
- **Centralizados**: Un problema que no se puede solucionar eficientemente al tener un sistema de control local, es la colaboración con otros desarrolladores. En respuesta se crearon sistemas de control (como CVS, Subversion y Perforce) en los que se comparte un servidor, que contiene los archivos versionados, entre todos los colaboradores de un proyecto.

La ventaja de estos sistemas frente a los locales, es la capacidad de poder averiguar quién se encuentra colaborando con qué parte del proyecto y poder controlar el acceso a sus partes. Además resulta más fácil administrar este tipo de sistemas que los sistemas locales debido a que estos últimos se encuentran esparcidos en varias máquinas (en caso de colaborar con otras personas).
La desventaja es consecuencia de la misma centralización. Al tener el proyecto completo ubicado en un solo servidor, una falla de acceso o (en el peor de los casos) una pérdida de datos, puede significar la pérdida parcial o total del proyecto, en caso de no haber una copia de respaldo del proyecto. En caso de tratarse de una pérdida de acceso provisoria, es posible que no se pueda seguir trabajando hasta recuperarlo.

- **Distribuidos**: En los sistemas de control de versionado distribuidos (como Git, Mercurial, Bazaar o Darcs) los clientes mantienen una copia total del proyecto en cuestión. De esta forma si el servidor llega a comprometerse, siempre se puede restaurarlo a partir de uno de sus clientes.

En el caso del proyecto de esta Tesina, se ha optado por utilizar Git, uno de los sistemas distribuidos más populares en la actualidad. A continuación se detallan algunas de sus características (ver 4.4.1).

Fuente

- 1.1 Getting Started - About Version Control. (n.d.). Consultado el 15 de Marzo de 2015, Obtenido de http://git-scm.com/book/en/v2/Getting-Started-About-Version-Control

4.4.1. Git

Git es un sistema de control de versionado distribuido, muy utilizado en la actualidad. Anteriormente se solía recurrir mayormente a sistemas centralizados como SVN (4.4.1.2). Al ser distribuido, Git permite almacenar una copia completa de un proyecto en cada dispositivo que trabaja con el mismo. Si bien esto significa ocupar mayor espacio en memoria, por otro lado se asegura que se pueda seguir trabajando en la totalidad del proyecto aún cuando no se tenga acceso al repositorio de origen. Además se aminora la probabilidad de perder el proyecto, al tener varias copias del mismo.
Git es compatible con una amplia variedad flujos de trabajo, de manera que es muy adaptable a la organización de diversos proyectos.
La característica más prominente de Git es el sistema de *branching* (ramificación), que permite crear líneas de desarrollo totalmente independientes. Por ejemplo, se puede crear una funcionalidad de forma experimental para una aplicación, sin tener que afectar la versión principal.

Una vez terminado el desarrollo correspondiente, se puede decidir si se desea agregar la funcionalidad a la línea de desarrollo principal o si se prefiere descartarla.
El branching también puede constituir una manera para separar el desarrollo de partes de una aplicación, entre varios desarrolladores que colaboran en un mismo proyecto. En tal caso se tendría un branch principal y otro por cada desarrollador que colabore. Cada uno de estos branches se puede 'fundir' (merge) con el el branch principal, una vez terminado su desarrollo.
Para el proyecto de esta Tesina se adoptó un workflow que consiste en 6 iteraciones. En el repositorio Git, esto se refleja como 6 branches, uno principal (master) y otros cinco para las iteraciones, desde la segunda hasta la sexta. La primer iteración se llevó a cabo directamente sobre el branch principal, dado que esta constituye la base para el resto del proyecto.

Fuente

- Git. (n.d.). Consultado el 15 de Marzo de 2015, Obtenido de http://git-scm.com/
- About. (n.d.). Consultado el 15 de Marzo de 2015, Obtenido de http://git-scm.com/about

4.4.1.1. GitHub

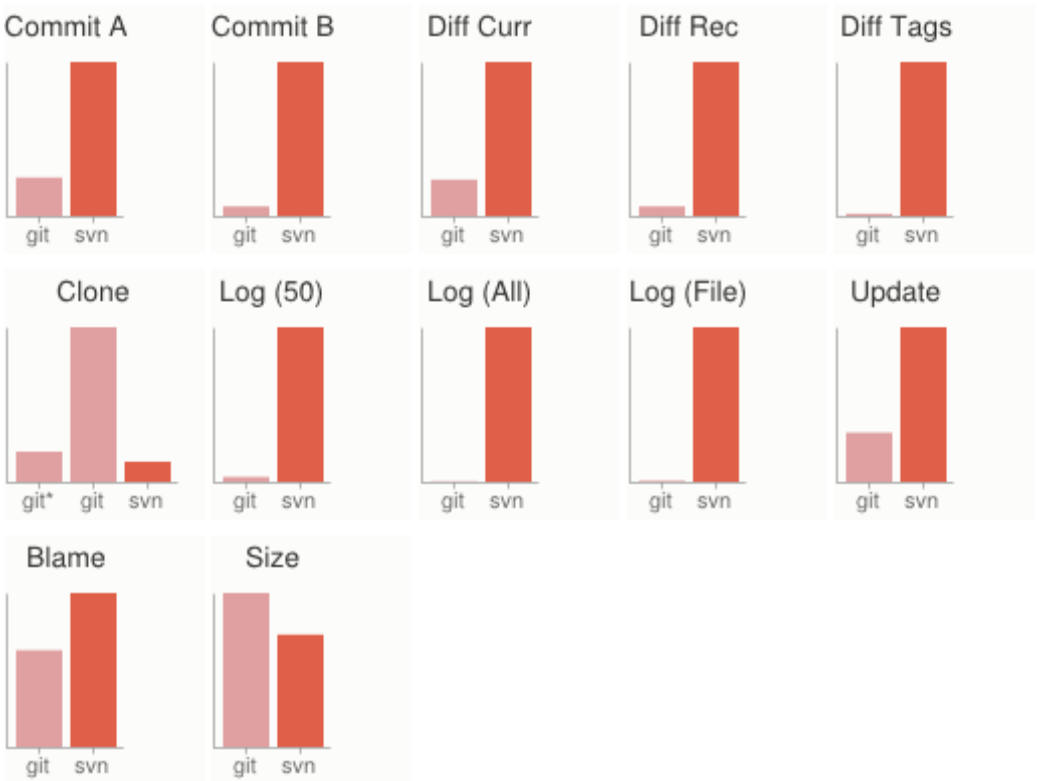
GitHub es la plataforma de repositorios de código más grande del mundo, con más de 20.6 millones de repositorios. Permite albergar proyectos en forma pública (gratuita) y privada (bajo costo mensual).
La plataforma provee una serie de herramientas, con énfasis especial en fomentar la colaboración entre desarrolladores. Una de estas es el *issue tracker* que permite reportar problemas (con atributos como por ejemplo etiquetas), que luego pueden ser asignados a colaboradores para ser resueltos. Se pueden establecer milestones para cumplir objetivos con plazo de vencimiento, referenciar versiones (*commits*), comparar cambios, y cerrar reportes de problemas, entre otras funcionalidades.
Otra herramienta muy útil, soportada directamente desde Git, son los *pull requests*. Esto son propuestas de cambios en el código que consisten en: El problema a solucionar o la característica a agregar, el código que soluciona la cuestión y comentarios sobre el código.
En caso que se contara con un equipo de desarrollo establecido, se puede crear equipos dentro de organizaciones. Cada uno de estos equipos puede tener asignados diferentes niveles de acceso y permisos.
A modo ilustrativo y con el fin de asegurar el código de este proyecto, se ha creado un repositorio en GitHub bajo el vínculo https://github.com/donboli/socialfeed. En este lugar se alberga la aplicación con acceso público.

Fuente

- Build software better, together. (n.d.). Consultado el 16 de Marzo de 2015, Obtenido de https://github.com/features

4.4.1.2. Comparación de Git con SVN

Las ventajas de Git frente a SVN (Subversion) resultan obvias considerando las diferencias entre un sistema de control de versionado distribuido frente a uno centralizado (ver 4.4).
Sin embargo, dichas diferencias no demuestran ventajas de eficiencia a nivel de operaciones. Es decir, de las ventajas genéricas de Git frente a SVN no se puede derivar que uno sea más rápido que el otro, sin mayores consideraciones. La comparación entre un sistema y el otro no es siempre directa, dado que algunas operaciones en SVN, como *commit*, suelen implicar un cambio directo en el repositorio del servidor, mientras que la misma operación en Git puede tener un efecto meramente local. A modo de ejemplo: *commit* en SVN equivale a *add* + *commit* + *push* en Git.
Por lo tanto, para los siguientes benchmarks se tuvo en cuenta operaciones equivalentes con Git frente a operaciones menos atómicas en SVN.



git*: *shallow clone*, trae únicamente la última versión del repositorio (de un sólo branch)

Las columnas representan el tiempo insumido por cada sistema en la operación correspondiente. Las condiciones de hardware y repositorio son iguales para ambos sistemas de versionado.

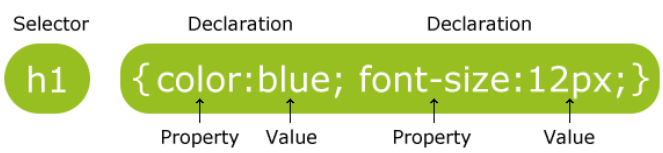
Como se puede apreciar, casi todas las operaciones resultaron más rápidas y por ende más eficientes (debido a la igualdad de recursos) en Git. Las únicas 2 operaciones que demandan mayor procesamiento en Git que en SVN son *clone* y *size*.
Para el caso de *clone* la razón está en que Git descarga el repositorio completo, con todo su historial de versiones. En cambio SVN sólo descarga la versión actual, quedando las demás en el repositorio central. Afortunadamente esta operación es de uso poco frecuente, considerando que se requiere sólo para iniciar el desarrollo en un dispositivo.
Lo anterior también justifica el rendimiento aparentemente inferior de *size* en Git, teniendo en cuenta que no se calcula únicamente el tamaño de la última versión, sino el total del repositorio.

Fuente

- About. (n.d.). Consultado el 16 de Marzo de 2015, Obtenido de http://git-scm.com/about/small-and-fast

4.5. CSS

CSS (Cascading Style Sheets) es un lenguaje para escribir hojas de estilo que definen la apariencia de una página web.
Se creó como consecuencia de las complicaciones surgidas a la hora de integrar estilos en HTML. Históricamente HTML se concibió con el único propósito de definir el contenido de una página y su estructura. Dicha finalidad se siguió con rigor hasta la aparición de HTML 3.2, en donde se incorporaron etiquetas y atributos que permiten especificar la apariencia del contenido. A partir de entonces el problema era especificar los estilos por cada elemento por separado y para cada una de las páginas. Es decir, se tenía que repetir constantemente los estilos para cada elemento que los debía adoptar.
Con HTML 4.0 se introdujo CSS (creado por el World Wide Web Consortium), con el fin de remover los estilos de los archivos HTML y definirlos en archivos de extensión ".css" externos. De esta forma la apariencia se pudo definir en un sólo lugar y aplicarse para todas las páginas de un sitio o aplicación web.
Con CSS se ahorra tiempo, se fomenta la reutilización de estilos y se facilita su mantenimiento.
Las hojas de estilo contienen un conjunto de reglas que identifican los elementos a los que se deben aplicar y declaran los estilos correspondientes. La sintaxis es la siguiente:



Cómo se puede ver en la ilustración, cada regla está formada por:

1. Un **selector** que identifica los elementos HTML que se deben estilizar.
2. Un **conjunto de declaraciones**, cada una de las cuales consta de una propiedad con su respectivo valor.

Fuente

- CSS Syntax. (n.d.). Consultado el 16 de Marzo de 2015, Obtenido de http://www.w3schools.com/css/css_syntax.asp

4.5.1. Sass/SCSS

CSS es una buena alternativa para extraer los estilos de los archivos HTML. Sin embargo, todavía se tiene que repetir mucho código dentro de las hojas de estilo, debido a su sintaxis restrictiva.
A raíz del problema anterior surgió una serie de alternativas que permiten declarar estilos de manera que se tenga que repetir menos código, y así volver más cortas, flexibles y mantenibles las hojas de estilo. Los 2 lenguajes alternativos más populares son Sass (Syntactically Awesome StyleSheets) y Less.
En la aplicación de este proyecto se utilizó Sass por razones de previa familiarización con dicho lenguaje.

SCSS es una extensión de CSS que permite el uso de variables, reglas anidadas, mixins (equivalente de funciones que se reemplazan por las propiedades contenidas), importaciones en línea, y otras funcionalidades, siendo totalmente compatible con la sintaxis de CSS. Lo último significa que la sintaxis de CSS es un subconjunto de SCSS.
Sass no se interpreta directamente en el cliente, sino que debe ser precompilado a CSS para luego ser enviado al cliente. Esto se realiza porque hasta el momento los clientes (navegadores principalmente) sólo soportan CSS como hojas de estilo. Sin embargo, se cumple el objetivo principal de facilitar y mejorar el manejo de las hojas de estilo.

Sass cuenta con 2 sintaxis:

1. **SCSS** (Sassy CSS): Es la extensión de CSS, con lo cual cualquier hoja de estilo CSS funcionará siendo procesada como SCSS. La extensión para los archivos de este tipo es ".scss".
2. **Sass**: Es la sintaxis vieja, conocida también como sintaxis indentada. Utiliza indentación en vez de llaves ("{}"), lo cual la vuelve más concisa pero incompatible con CSS plano. Los archivos de este tipo llevan la extensión ".sass".

Fuente

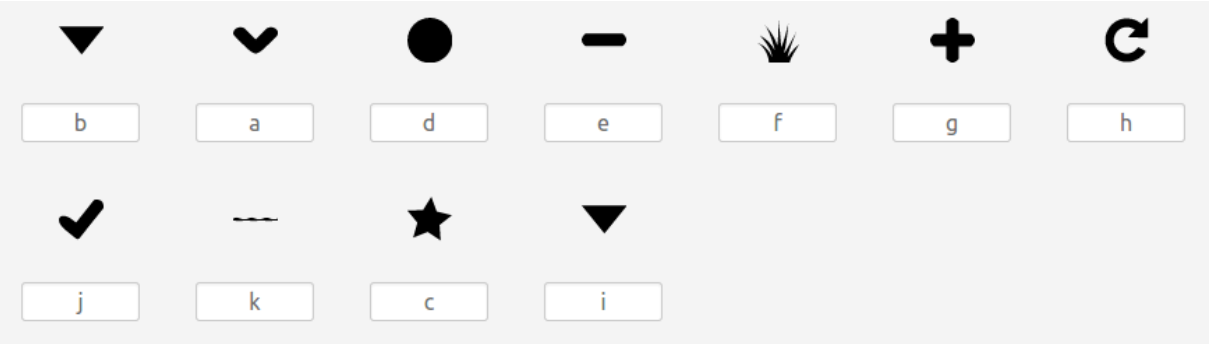
- Sass (Syntactically Awesome StyleSheets). (n.d.). Consultado el 16 de Marzo de 2015, Obtenido de http://sass-lang.com/documentation/file.SASS_REFERENCE.html

4.5.2. Fontastic

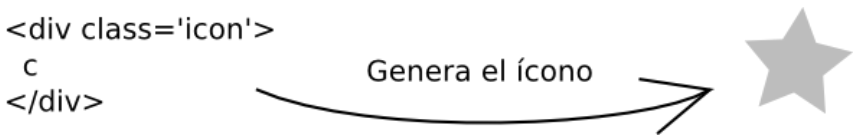
Los íconos e imágenes que se usan en este proyecto son exclusivamente vectoriales. Esto trae como ventaja que al cambiar su tamaño no cambie la nitidez, sino que la calidad persista y se reajuste la imagen automáticamente.
En contraposición, en muchas aplicaciones se utilizan imágenes rasterizadas, y para poder mantener la calidad se recurre a varias copias de una misma imagen con resoluciones diferentes. Esto requiere el uso de media queries, que son un tipo de regla de CSS que aplica estilos, según si se cumplen ciertas condiciones del dispositivo de salida. En concreto, cuando se usa imágenes rasterizadas es necesario realizar un sondeo del tamaño de pantalla, para saber qué resolución debe tener la imagen que se muestra.
Usar imágenes vectoriales permite dejar de lado los detalles anteriores. Sin embargo, se debe tener cuidado con el tamaño de las imágenes vectoriales, cuando su complejidad es muy grande.

Tanto en el caso de imágenes vectoriales como rasterizadas, no se puede modificar la apariencia de éstas mediante CSS. Por ejemplo, si se quisiera cambiar el color de una imagen y agregarle una sombra, esto no sería posible. Al menos, esto es el caso cuando se encapsula las imágenes en un elemento *img* de HTML.

Una modalidad para poder aplicar estilos a imágenes vectoriales es creando un *font* (tipo de letra) a partir de un archivo que contenga dichas imágenes.
Para facilitar esto existen diversas aplicaciones. Para el presente proyecto se ha utilizado **Fontastic**. Esta herramienta web permite subir iconos en formatos vectoriales, como SVG (Scalable Vector Graphics), asignarle a cada ícono una letra, y luego publicar o exportar un tipo de letra que los contenga.
Los íconos de este proyecto fueron creados en su mayoría con **Inkscape**, una herramienta de edición de gráficos vectoriales, similar a Adobe Illustrator. Una vez exportados en formato SVG, se subieron a Fontastic para obtener los iconos asignados a un tipo de letra. A continuación, una captura de pantalla de Fontastic con los íconos y sus respectivas letras.



En la aplicación, estos iconos son referenciados por un elemento *div* o *span* con clase “icon” y como contenido una letra correspondiente al ícono que se quiera visualizar. En el código esto sería por ejemplo:



Fuente

- Add Vector Icons to Your Website. (n.d.). Consultado el 17 de Marzo de 2015, Obtenido de http://fontastic.me/

4.5.3. Responsive Design

Una página web tiene un diseño adaptable o *responsive design* cuando responde con diferentes disposiciones de sus vistas, según el tamaño de la ventana. Generalmente esto significa que según el dispositivo se vea diferente. El fin es atender una amplia gama de dispositivos de forma óptima.

Los elementos a los que se recurre normalmente son: cuadrículas fluidas o flexibles, imágenes fluidas y Media Queries de CSS.

Rara vez se puede dar por concluido el diseño, dado que constantemente aparecen nuevos dispositivos en el mercado que requieren un ajuste para poder visualizarse bien las páginas.

Crear un diseño adaptable sin una base preexistente puede ser muy laborioso y demandar un amplio conocimiento de HTML, CSS y JavaScript. Por suerte, en la actualidad existen frameworks de desarrollo web que se enfocan en solucionar esta problemática. Los más comunes son Bootstrap, Foundation y en menor medida Skeleton.

Para la aplicación de esta Tesina se eligió Bootstrap, por contar con experiencias previas en su uso. Por limitaciones de tiempo y para no desviarse del desarrollo funcional, la aplicación solo cuenta con responsive design en el landing (pantalla inicial de la aplicación), con el fin de ejemplificar esta característica.

Fuente

- What is Responsive Design? (n.d.). Consultado el 17 de Marzo de 2015, Obtenido de http://www.ugurus.com/responsive-design-definition

4.5.3.1. Bootstrap

Bootstrap fue creado en 2010, originalmente como framework de base en Twitter. Luego se liberó al público como software de código abierto y actualmente es uno de los frameworks de desarrollo front-end más populares del mundo.
Desde la segunda versión de Bootstrap se incluye soporte para responsive design. A partir de Bootstrap 3 el framework brinda responsive design por defecto con enfoque en la metodología *mobile first*. Esta última fomenta el diseño de aplicaciones desde un punto de partida de dispositivos móviles, para luego avanzar a aspectos adicionales. La metodología obliga a identificar y desarrollar primero las funcionalidades más básicas de una aplicación, dadas las restricciones de espacio en los dispositivos móviles. Posteriormente se agregan características avanzadas para ser utilizadas en otros dispositivos.
Las hojas de estilo de Bootstrap están escritas con el preprocesador Less. Como se mencionó previamente en 4.5.1, se trata de un preprocesador que se traduce en CSS. Sin embargo, también se puede descargar su variante en Sass (SCSS).
Como muchos otros frameworks front-end, Bootstrap consta de un sistema de cuadrillas de 12 columnas, con un espacio entre cada par. La distribución de las columnas depende del tamaño de la ventana y puede variar por medio de puntos de quiebre, que cambian la disposición de los elementos en la página. En el inciso 5.4.6.3 se ilustran estos cambios en la apariencia de la página landing.
Los componentes más importantes que conforman el framework son: Botones, Dropdowns, barras de navegación, botones de paginación, etiquetas y medallas, tipografías, alertas, modals, barras de progreso, entre otros. Además contiene un conjunto de más de 250 iconos vectoriales gratuitos, que se utilizan mediante un tipo de letra (como se explicó en 4.5.2).

Se concluye de esta forma el marco tecnológico del presente trabajo. En la sección siguiente se detalla el desarrollo de la aplicación, desde la definición de requerimientos básicos, hasta el término de la última iteración.

Fuentes

- Bootstrap · The world's most popular mobile-first and responsive front-end framework. (n.d.). Consultado el 17 de Marzo de 2015, Obtenido de http://getbootstrap.com/
- About. (n.d.). Consultado el 17 de Marzo de 2015, Obtenido de http://getbootstrap.com/about/

5. Desarrollo

5.1. User Stories

Uno de los formatos más comunes para definir user stories, que se utilizará para escribir user stories, es el siguiente: Como (rol) quiero (algo) para poder (beneficio).
Se recurre a esta técnica por constituir una manera simple de definir requerimientos. En esta parte no se pretende detallar la interacción precisa entre la aplicación y el usuario, sino establecer un punto de partida y base mínima de funcionalidades. Algunos detalles se definen implícitamente en el diseño de la interfaz gráfica, otros a lo largo del desarrollo de las iteraciones.
En forma simplificada, la aplicación a desarrollar se basa en la asociación de feeds con usuarios, de manera que cada usuario pueda mantener un registro de las fuentes de noticias (entradas). Entre estas dos entidades se encuentran las categorías, que sirven para agrupar las fuentes, según las convenciones del usuario. Por lo tanto las funcionalidades básicas están enfocadas en el manejo de estas 3 entidades: Usuario, Categoría y Feed.

Usuarios:

1. Signup de usuario: Como visitante quiero crear un usuario para guardar mis categorías, feeds, notificaciones y filtro de manera que pueda volver a acceder a estos en el futuro.
2. Login de usuario: Como visitante registrado quiero conectarme a mi usuario para acceder a la información guardada.
3. Logout de usuario: Como usuario quiero desconectarme para impedir el acceso indiscriminado y permitir a otros visitantes acceder a sus respectivos usuarios desde la misma máquina.
4. Baja de usuario: Como usuario quiero eliminar mi cuenta para poder borrar toda información personal o metadatos creados por mí.

Categorías:

1. Crear Categoría: Como usuario quiero crear una categoría proveyendo un nombre y opcionalmente una descripción y etiquetas, para poder agrupar los feeds según criterios personales.
2. Modificar Categoría: Como usuario quiero poder modificar el nombre, descripción y etiquetas de una categoría para poder adaptar la información que representa a los feeds asociados.
3. Eliminar Categoría: Como usuario quiero poder eliminar una categoría para poder deshacerme rápidamente del grupo de feeds asociado.

Feeds:

1. Agregar Feed: Como usuario quiero agregar un feed para poder obtener las entradas correspondientes.
2. Eliminar Feed: Como usuario quiero eliminar un feed con el fin de no seguir recibiendo entradas o notificaciones correspondientes.

Las entradas de los feeds se deben poder manejar y filtrar de diferentes formas para que la información sea fácilmente accesible, organizable y poder presentarse en forma útil para el usuario. Los siguientes user stories son referentes a las entradas de los feeds, los filtros y las notificaciones.

Entradas:

1. Marcar como leída: Como usuario quiero marcar una entrada como leída para poder mantener el rastro de las entradas que ya leí.
2. Marcar como no leída: Como usuario quiero marcar entradas leídas como no leídas con el fin de volver a leerlas más tarde.
3. Marcar como favorita: Como usuario quiero marcar entradas como favoritas para poder guardarlas permanentemente y acceder a estas en cualquier momento futuro.
4. Desmarcar favorita: Como usuario quiero desmarcar una entrada favorita para deshacerme de esta.
5. Filtrar por categoría: Como usuario quiero filtrar las entradas por categorías para poder acotar las entradas según un conjunto específico de feeds.
6. Filtrar por feed: Como usuario quiero filtrar las entradas por feeds específicos para poder excluir información que no resulta de interés en el momento. En caso de estar filtrando por categoría quiero que se pueda filtrar únicamente por los feeds pertenecientes a la categoría seleccionada.
7. Filtrar por leídos/favoritos: Como usuario quiero filtrar las entradas por leídas/no leídas y favoritas/no favoritas para poder organizar mejor la información.

Filtros:

1. Crear Filtro: Como usuario quiero crear un filtro para un feed particular para poder aminorar la cantidad de entradas que recibo según palabras clave que contienen.
2. Modificar Filtro: Como usuario quiero modificar un filtro para un feed para poder ajustar o cambiar el criterio mediante el cual se excluyen entradas que no resultan de interés.
3. Eliminar Filtro: Como usuario quiero eliminar un filtro para un feed con el fin de evitar que se sigan excluyendo entradas.

4. Desactivar Filtro: Como usuario quiero desactivar un filtro para un feed con el fin obtener temporalmente todas las entradas asociadas, sin tener que eliminar el filtro.
5. Activar Filtro: Como usuario quiero activar un filtro para un feed para poder recibir sólo aquellas entradas que cumplan el criterio establecido.

Notificaciones:

1. Crear Notificación: Como usuario quiero crear una notificación para poder recibir alertas vía mail cuando aparezca una entrada de un feed particular que contenga alguna palabra específica.
2. Modificar Notificación: Como usuario quiero modificar una notificación para cambiar los criterios por los cuales recibo alertas vía mail.
3. Eliminar Notificaciones: Como usuario quiero eliminar una notificación para dejar de recibir alertas vía mail permanentemente.
4. Desactivar Filtro: Como usuario quiero desactivar un filtro para un feed con el fin obtener temporalmente todas las entradas asociadas, sin tener que eliminar el filtro.
5. Activar Filtro: Como usuario quiero activar un filtro para un feed para poder recibir sólo aquellas entradas que cumplan el criterio establecido.

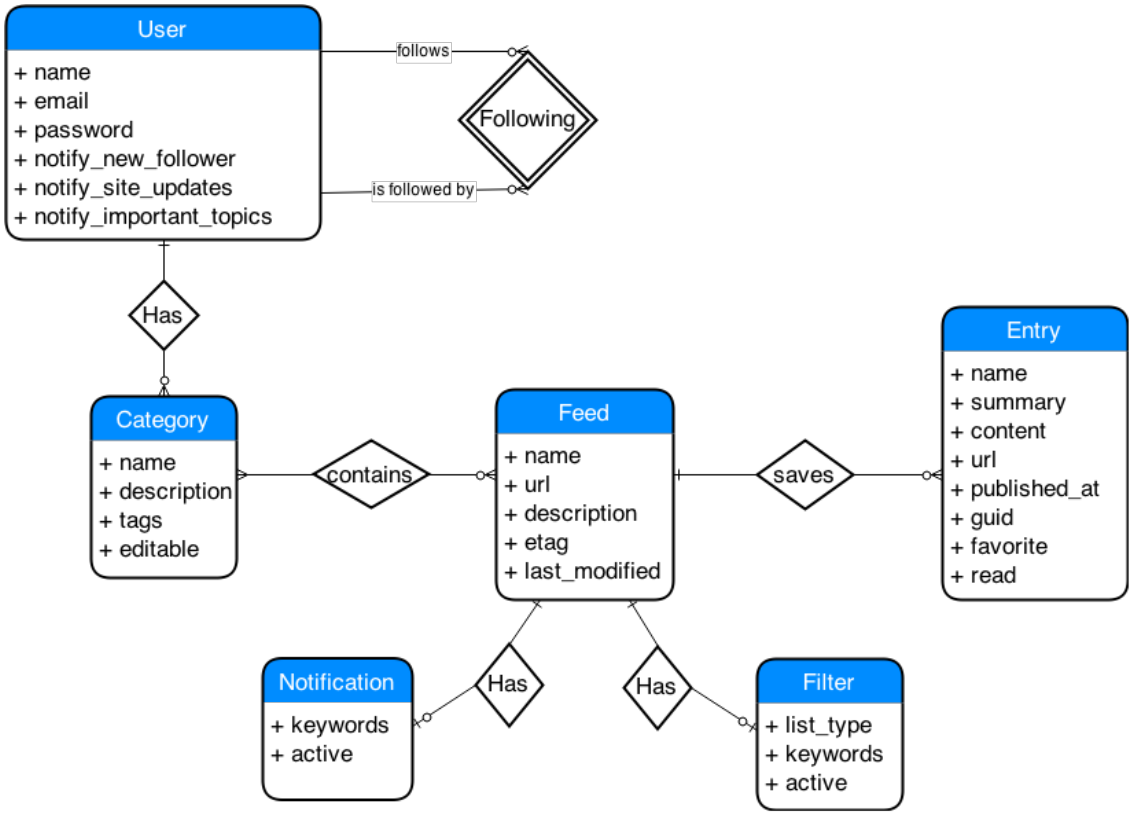
Por último se debe poder agregar otros usuarios a la cuenta actual, de manera que se pueda acceder a las entradas de estos en forma fácil en todo momento.
Asociación con otros usuarios:

1. Asociar usuario: Como usuario quiero agregar otros usuarios a un listado para tener acceso rápido a las entradas que recibe el usuario seleccionado.
2. Desvincular usuario: Como usuario quiero desvincular usuarios de mi cuenta para dejar de tener un acceso directo a sus entradas.

Fuente

- Writing user stories. (n.d.). Consultado el 17 de Marzo de 2015, Obtenido de https://www.gov.uk/service-manual/agile/writing-user-stories.html

5.2. Diagrama Entidad-Relación (DER)



La mayoría de los atributos no requieren de explicaciones porque su significado se volverá evidente a medida que se revisa el desarrollo de la aplicación. Otros se requieren por detalles técnicos con poca relevancia a nivel de negocio. Estos últimos se explican brevemente a continuación.

- **Feed** tiene 2 atributos con la misma finalidad, aunque su funcionamiento es sutilmente diferente. Estos son ***etag*** y ***last_modified***. Ambos existen para optimizar el uso del ancho de banda, evitando descargas cuando no hubo cambios desde la última actualización del feed.

  El atributo *etag* (entity tag) proporciona un código de identificación del feed, que cambia según su estado. Cada vez que hay un cambio en el feed, el etag cambia y por lo tanto, al comparar el etag viejo con el nuevo, se detecta si hay entradas nuevas para descargar.

  El atributo *last_modified* contiene simplemente una marca de tiempo de cuando se realizó la última modificación al feed. Comparando la marca del feed con su último registro en la base de datos se puede determinar si es necesario descargar nuevas entradas.
- **Entry** cuenta con el atributo ***guid*** (globally unique identifier) para poder identificar unívocamente cada entrada. La razón para esto es evitar entradas duplicadas en la base

de datos. Se debe tener esta precaución debido a que, cuando se descargan entradas de un feed, se tiende a descargar entradas anteriormente grabadas, dependiendo de la frecuencia de consulta.

Fuente

- ETag and Last-Modified Headers¶. (n.d.). Consultado el 18 de Marzo de 2015, Obtenido de https://pythonhosted.org/feedparser/http-etag.html

5.3. Interfaz Gráfica de Usuario

Para armar el diseño de la aplicación se utilizó Inkscape, un editor de gráficos vectoriales de código abierto, similar a Adobe Illustrator. Ante todo se seleccionó este programa por ser gratuito, además de contar con Linux como plataforma principal. Además, permite trabajar con la especificación SVG (Scalable Vector Graphics) en forma nativa. Esto último posibilita la exportación de imágenes vectoriales a archivos con formato XML, que pueden ser utilizados directamente en una aplicación web. Las ventajas de este formato son principalmente:

1. Ajuste automático de tamaño de imagen sin pérdida de calidad.
2. Menor tamaño de archivos en caso de imágenes simples*.

* En el caso de gráficos vectoriales una imágen es más simple cuanto menos vectores requiere para ser descrita.

Para definir el esquema de colores del diseño, se utilizó una aplicación web conocida como Paletton (www.paletton.com). Con “esquema” se refiere al grupo de colores que conforman los elementos de la página web. Es decir, se determinó de antemano un conjunto de colores que se podrán usar en el encabezado, barra lateral, pie de página y contenido, así como todos los demás elementos visibles. Seleccionar un buen esquema puede tener efectos sobre el ánimo de los usuarios, pero además es importante lograr que la página resulte atractiva y combine con el propósito del sitio. En el caso de diseñadores experimentados, es posible que se componga intuitivamente el esquema. Dado que no se cuenta con los conocimientos ni experiencia suficiente para tomar decisiones de este tipo sin asistencia externa, se optó por utilizar Paletton. El esquema resultante se muestra a continuación.

En los siguientes incisos (5.3.1 a 5.3.6) se muestran los diseños generados con Inkscape. No se expondrán detalles acerca del proceso que llevó a crear las diferentes vistas, debido a que el presente trabajo se enfoca en aspectos de implementación y no de diseño gráfico.

Importante: Las imágenes que se muestran a continuación no corresponderán con exactitud con aquellas de la aplicación final, debido a ciertas limitaciones, ya sean técnicas o de conocimiento. Compárese por ejemplo el diseño de la vista landing con las versiones finales en 5.4.6.3.

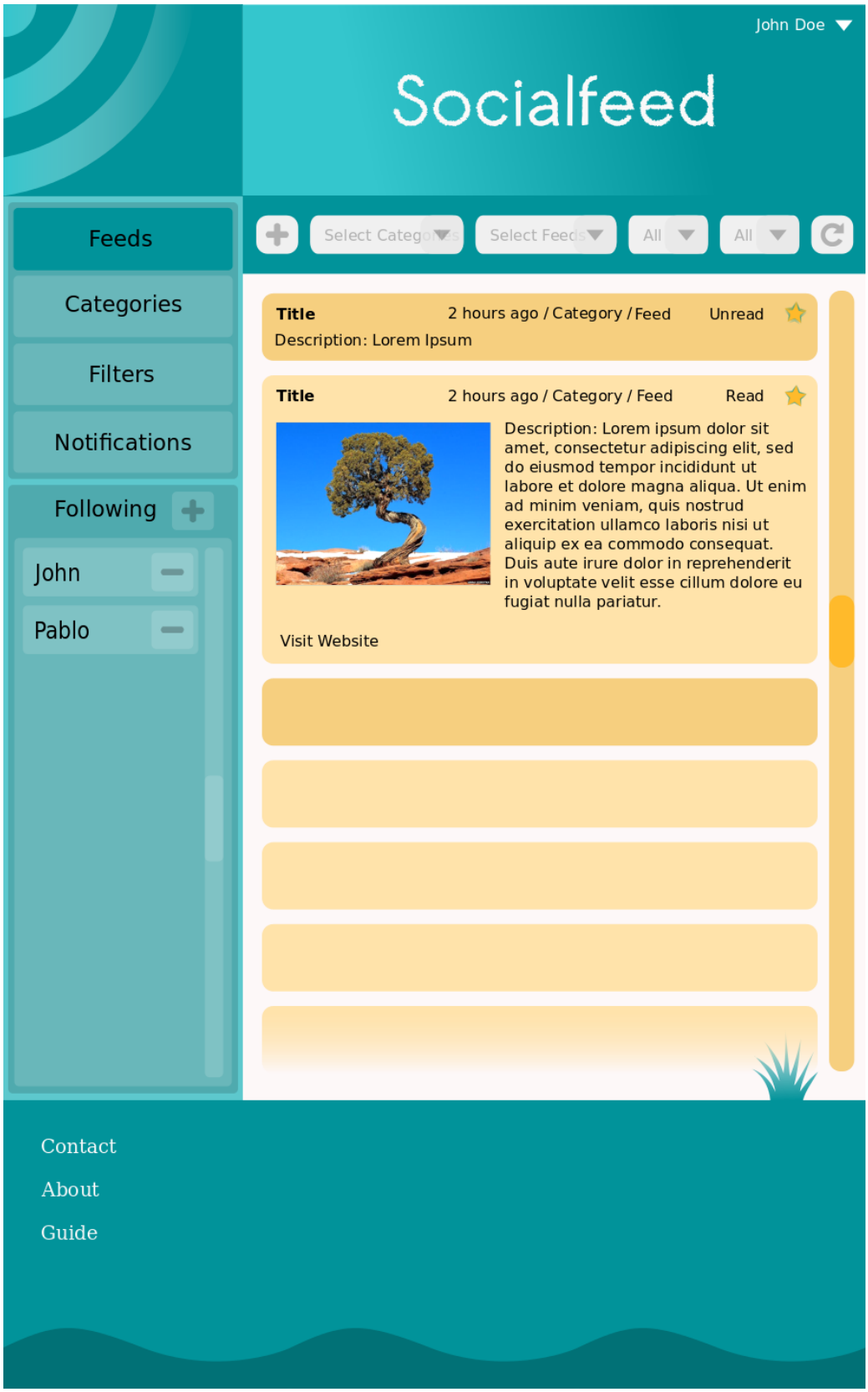
Fuente

- Usage. (29 de Agosto de 2011). Consultado el 18 de Marzo de 2015, Obtenido de http://www.colorschemedesigner.com/blog/usage/

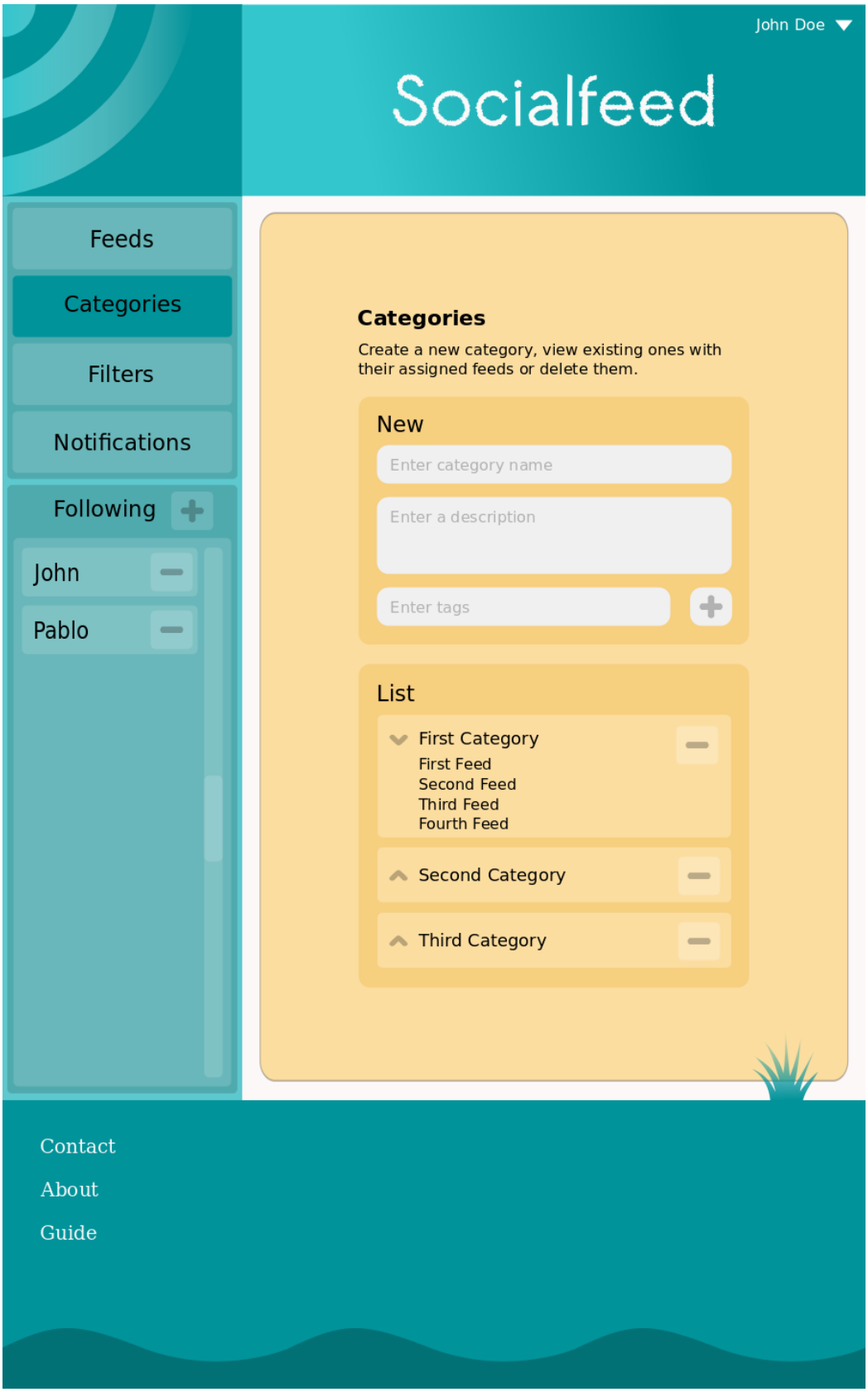
5.3.1. Landing


Socialfeed
Keep up with your interests and those of whom you care about
Sign Up
Email
Pick a username
Re-enter email
Password
Sign Up!
Already got an account?
Email
Password
Sign In
Store feeds in a single place
Stay informed
Share with others
Contact
About
Guide
2014 MIT License

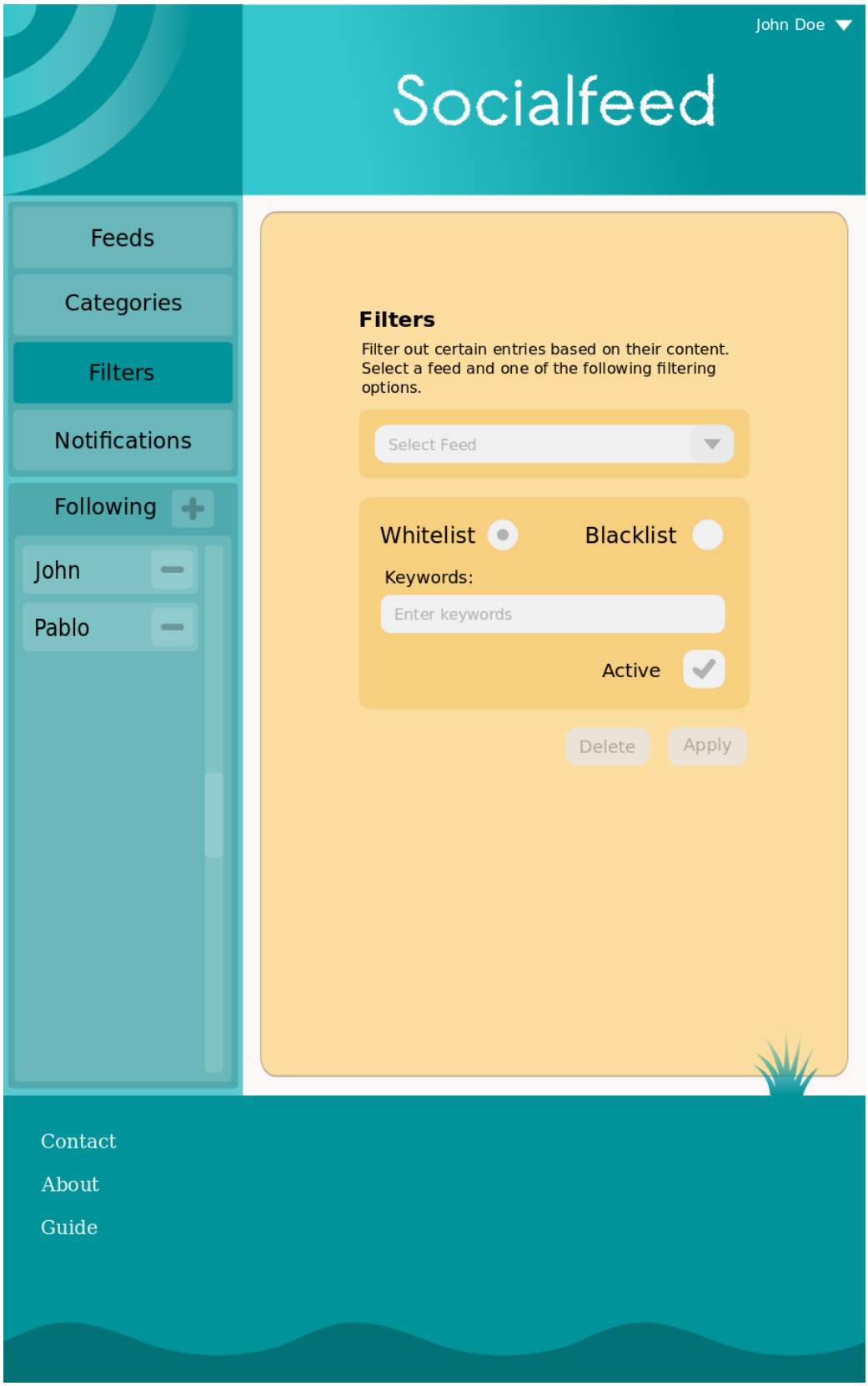
5.3.2. Feeds (Home)



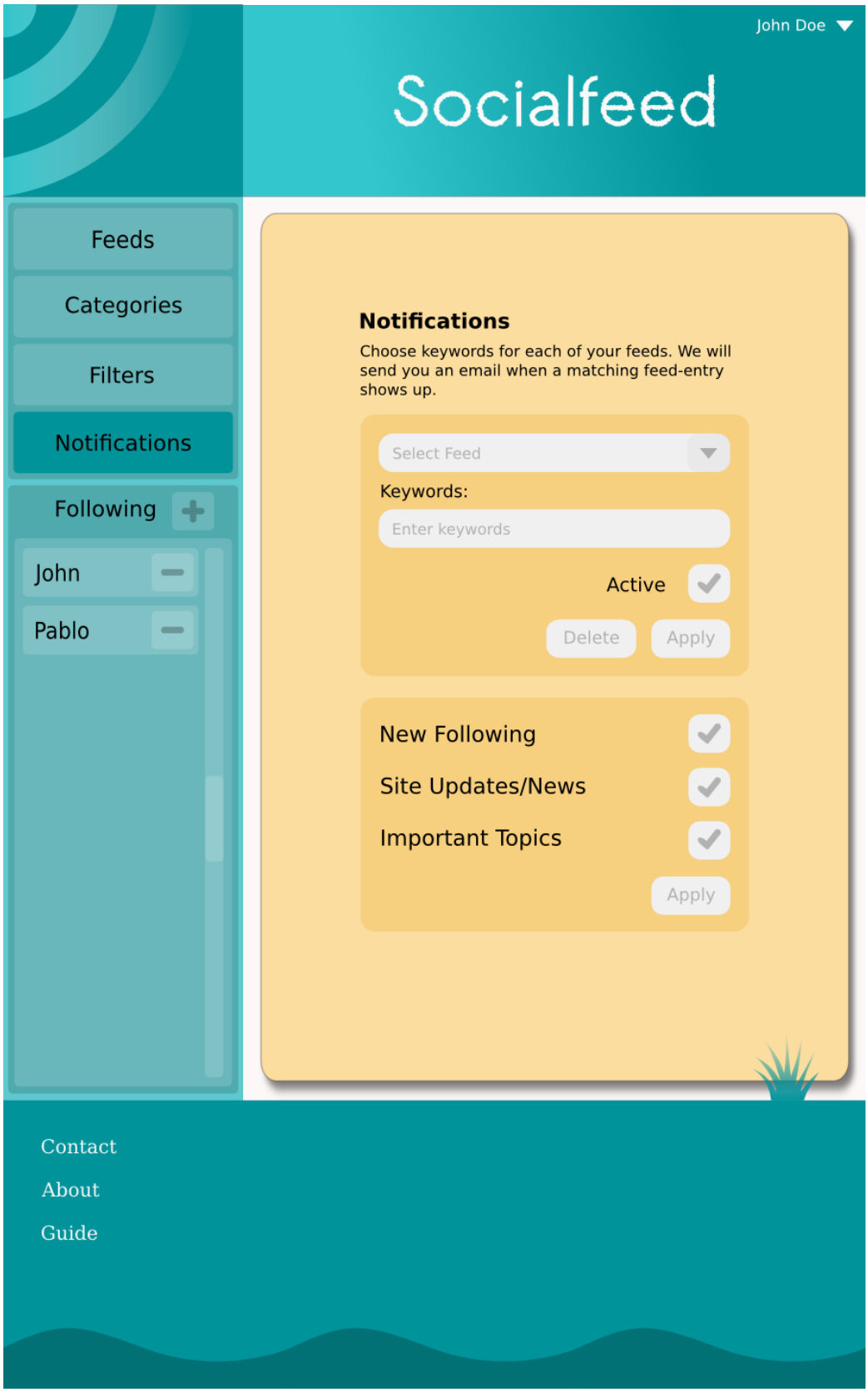
5.3.3. Categorías



5.3.4. Filtros


John Doe
Socialfeed
Feeds
Categories
Filters
Notifications
Following
John
Pablo
Filters
Filter out certain entries based on their content.
Select a feed and one of the following filtering options.
Select Feed
Whitelist
Blacklist
Keywords:
Enter keywords
Active
Delete
Apply
Contact
About
Guide

5.3.5. Notificaciones



5.3.6. Seguir usuarios

5.4. Iteraciones

Las iteraciones constituyen una manera de desglosar el desarrollo de una aplicación en conjuntos de características y funcionalidades que pueden ser diseñados, desarrollados y testeados en ciclos repetidos. En el caso de este proyecto las iteraciones son las siguientes:

1. **Iteración 1 (Funcionalidad básica)**: ABM de usuarios, y agregado y visualización de feeds bajo una categoría genérica (indefinida). Actualización de entradas de feeds.
2. **Iteración 2 (Categorías y visualización avanzada)**: ABM de categorías, eliminación de feeds y filtrado de entradas en la visualización.
3. **Iteración 3 (Filtros)**: ABM de Filtros.
4. **Iteración 4 (Notificaciones)**: ABM de Notificaciones.
5. **Iteración 5 (Seguir usuario)**: Vinculación/desvinculación de usuarios y visualización de entradas correspondientes.
6. **Iteración 6 (Landing)**: Maquetado de landing, notificaciones flash generales, responsive design para landing.

Importante: Se han omitido explicaciones acerca de la instalación y configuración de Rails y sus dependencias, por tratarse de una tarea genérica que no depende de la aplicación particular. El proceso que se siguió se encuentra detallado en: https://gorails.com/setup/ubuntu/14.04

Fuente

- What is iterative development? - Definition Obtenido de WhatIs.com. (n.d.). Consultado el 17 de Marzo de 2015, Obtenido de http://searchsoftwarequality.techtarget.com/definition/iterative-development

5.4.1. Iteración 1

Para crear una nueva aplicación con Ruby on Rails se debe abrir una terminal de Linux (en el caso particular se utiliza Ubuntu 14.04), navegar a la carpeta donde se quiera crear la aplicación, e ingresar el siguiente comando: **rails new socialfeed**, siendo 'socialfeed' el nombre de la aplicación. A continuación se crea automáticamente el sistema de archivos y se instalan las gemas que forman parte de la aplicación por defecto.
Una vez realizado esto, se puede comenzar con el desarrollo.

5.4.1.1. ABM de usuarios

El agregador RSS de este proyecto permitirá crear sesiones de usuario, bajo las cuales se podrán guardar las categorías y fuentes preferidas de cada usuario. Resulta esencial contar con el modelo de usuario para poder separar estas y otras entidades. El modelo de usuarios (User) se creará por medio de una gema llamada **Devise** (ver 4.1.1.2), que automatiza todos los aspectos de autenticación relevantes para este proyecto. Devise se ocupa de la autenticación mediante contraseñas encriptadas, el registro de usuarios, recuperabilidad de cuentas, memorización de sesiones por cookies, rastreo de sesiones, y validaciones.
Para todo lo anterior sólo se tienen que seguir los siguientes pasos:

1. Incluir la declaración **gem 'devise'** en el archivo Gemfile (ubicado en la raíz del proyecto) y correr **bundle install** en la terminal (shell).
2. Correr el generador **rails generate devise:install** en la terminal.
3. Ejecutar el comando **rails generate devise User**, siendo User el modelo de usuario. Esto crea el modelo con los módulos por defecto, genera la migraciones necesarias para el esquema de base de datos, crea todas las vistas necesarias y configura las rutas en routes.rb.
4. Finalmente se deben correr las migraciones con **rake db:migrate** (ver 4.1.1.5 para el concepto de migraciones).

A partir de este momento las peticiones a la aplicación (por defecto a la URL http://localhost:3000/) serán dirigidas a la vista de creación de usuario (si no está autenticado todavía) o al índice de home (pantalla principal de nuestra aplicación). De esto se ocupan las rutas que fueron generadas automáticamente.
Se ejecuta el comando **rails generate devise:views** para poder personalizar el diseño de las vistas más adelante. Será necesario ajustarlas a la hora de maquetar la vista de landing (ver 5.4.6).
Con esto se tiene una buena base para seguir con el desarrollo del núcleo de la aplicación: La gestión de fuentes RSS/Atom.

5.4.1.2. Creación de modelos

Mediante el uso de la gema de HoboFields (ver 4.1.1.5) se crean los modelos restantes, que figuran en el Diagrama de Entidad-Relación (ver 5.2).
Los comandos (generadores) correspondientes son:

```
$ rails g hobo:model category
...
$ rails g hobo:model entry
...
$ rails g hobo:model notification
...
$ rails g hobo:model filter
...
$ rails g hobo:model feed
...
```

En cada caso se genera un modelo en el directorio models/. Además se crean 2 archivos de testing bajo el directorio de spec/, uno para probar el modelo bajo spec/models/ y el otro para generar datos de prueba con el mismo fin (en spec/factories).
Esto es conveniente para poder probar cada modelo a medida que se implementa, y así mantener la integridad de la lógica de negocio, de la cual se deben ocupar los modelos. Los datos de prueba son usados directamente por **RSpec**.
Rails viene por defecto con otra utilidad para datos de prueba, conocida como **fixtures**. El problema de esta última es que no provee flexibilidad para representar asociaciones complejas entre diferentes modelos. Además es menos versátil para generar datos aleatorios. Por se opta por usar **FactoryGirl**, que permite crear datos en tiempo de ejecución y con parámetros flexibles.
A lo largo de este proyecto no se realizará testing por cuestiones de tiempo y recursos. Sin embargo, se generarán en forma implícita los directorios y archivos que permiten llevar a cabo testing en un futuro.
Originalmente Rails viene con generadores para crear migraciones. Estas últimas nos facilitan el manejo de la correspondencia entre los campos del modelo y la base de datos. Hacen posible manipular la base de datos programáticamente utilizando Ruby. La idea detrás de esto es mantener la definición de la base de datos fuera de un gestor de base de datos (DBMS) particular, con el fin de agilizar la creación y evolución del esquema de la base de datos, además de mantenerlo independiente de un gestor específico.

Si no se contara con esta característica se tendría que crear la base de datos mediante SQL en forma nativa. Significaría tener que generar un conjunto de sentencias SQL para un DBMS específico y cambiar este cada vez que se quisiera modificar la base de datos. Esto insumiría mucho tiempo, sin siquiera considerar la posibilidad de tener que cambiar el DBMS subyacente, en cuyo caso se estaría obligado a reescribir buena parte del código SQL, dado que este varía entre diferentes gestores. Una ventaja adicional es que las migraciones se generan en orden cronológico y se basan sucesivamente sobre las anteriores, manteniendo un historial del esquema.
HoboFields va un paso más allá, logrando que las migraciones se generen en forma automática a partir de la definición del modelo. Es decir, en vez de crear los campos manualmente mediante migraciones en la base de datos, sin dejar referencias en el modelo, los campos se definen en el modelo de forma explícita y luego se corre un generador para que cree de manera interactiva las migraciones. Así queda constancia de todos los campos en los mismos modelos, en vez de quedar únicamente en el esquema de base de datos (en el archivo db/schema.rb), y prácticamente uno se puede olvidar de que existen las migraciones.
A continuación se muestra un ejemplo de la definición de los campos del modelo Feed (en app/models/feed.rb):

```
fields do
  name        :string
  url         :string
  description :text
  timestamps
end
```

Luego las migraciones se pueden crear y correr interactivamente a través del comando:

```
$ rails g hobo:migration
```

Hecho esto sólo queda definir la relación de seguidores (followers) para el modelo de usuarios. Esta es la relación más compleja que se tendrá que definir en esta aplicación. Se trata de una relación de muchos a muchos pero auto-referenciada. Al ser un problema muy conocido, especialmente en la construcción de redes sociales, podemos recurrir a numerosas fuentes que proponen soluciones. En este caso encontramos una en Railscasts #163 que propone crear un modelo aparte para representar la relación. De esta manera no sólo se llega a una solución relativamente simple, sino que además se asegura una buena adecuación al estilo de arquitectura REST (Representational State Transfer), que es incentivado por Rails. Esto último se debe a que

con muchas otras soluciones se terminaría desobedeciendo al buen uso de los verbos HTTP, al asociar la gestión de la autoreferencia al propio modelo de usuario. Con la solución propuesta se puede tratar a la relación de seguimiento entre usuarios como una entidad aparte que se puede crear, ver, actualizar y destruir. Una de las restricciones que conforman REST es justamente mantener una interfaz uniforme basada en la manipulación de recursos. Cada verbo HTTP (a saber GET, POST, PUT, PATCH, DELETE) se asocia, bajo la restricción anterior, a una operación sobre el recurso en cuestión. Por ejemplo, GET trae un usuario de la base de datos, mientras que DELETE lo borra.
Por lo tanto, se crean otro modelo (Following) y las asociaciones entre este y el modelo de User como sigue:

Following:

```
  belongs_to :user
  belongs_to :followed, class_name: 'User'
```

User:

```
  has_many :followings
  has_many :followeds, through: :followings
  has_many :inverse_followings, class_name: "Following", :foreign_key =>
"followed_id"
  has_many :inverse_followeds, through: :inverse_following, :source =>
:user
```

Finalmente se corren de nuevo las migraciones. Con esto se concluye la definición de las relaciones entre modelos, para poder crear la primer vista.

Fuente

- Using HTTP Methods for RESTful Services. (n.d.). Consultado el 20 de Marzo de 2015, Obtenido de http://www.restapitutorial.com/lessons/httpmethods.html

5.4.1.3. Gestión de fuentes

Antes de poder trabajar con las fuentes (feeds) hace falta explicar algunos campos de la clase Entry.
Feed es la clase que representará a una fuente de información. Esto puede ser *RSS*, *Atom* o cualquier otro formato que sea soportado por la gema Feedjira (ver 4.1.1.3). Cada Feed tendrá que ser parte de una categoría (Category). Debido a dicha restricción habrá una instancia de

Category 'indefinida' que albergará a todos los feeds que no tengan una categoría asignada explícitamente. Esto sirve para agilizar el agregado de feeds, cuando no existen categorías.
Por otra parte se tiene la clase Entry, que representa un artículo o noticia particular de un feed. Es así como un feed puede tener asociadas cualquier cantidad entera positiva de registros de Entry. Las instancias de Entry se consiguen mediante Feedjira, que se ocupa de descargar los archivos xml correspondientes desde su origen y luego analizarlos (*parsing*) para crear las instancias de Feed. Una vez llevado a cabo esto último, se puede acceder al conjunto de entradas del feed.
Para nuestra aplicación se utiliza uno de los enfoques mostrados en el tutorial de Railscast #168 con el fin de almacenar las entradas de los feeds en nuestra base de datos, cada vez que estos se descargan. El atributo *GUID* (Globally Unique Identifier) de Entry, permite identificar a cada una de las entradas con el fin de poder evitar duplicados. Al crear instancias de Entry en la base de datos se tendrá en cuenta el GUID. Además, se guardará el atributo published_at con el fin de poder visualizar las entradas en forma ordenada, según su fecha y hora de publicación en el feed correspondiente.

Se agrega el atributo *etag* al modelo Feed para poder determinar si un feed se ha actualizado. Otro atributo que se agrega es *last_modified*, que contiene la fecha desde la última modificación del feed. El propósito de este campo es el mismo que de *etag*. Estos atributos cambian cuando se agregan nuevas entradas a la fuente, con lo cual se podrá detectar si es necesario realizar una descarga. Si esto no fuera el caso, se ahorra el tiempo requerido en descargar el feed completo y analizarlo (ver 5.2 para más información acerca de estos atributos).
Esta funcionalidad es más avanzada, por lo cual se implementará más adelante. Sin embargo, se dejará asentado el campo en la base de datos, para tenerlo en cuenta más adelante.

5.4.1.3.1. Actualización de feeds en Background

Las entradas de los feeds deben ser actualizadas continuamente, incluso cuando el usuario no esté conectado a su sesión. Esto se debe a que de otra manera el usuario no recibiría las notificaciones por email cada vez que se cumpla el criterio de alerta. Sólo las recibiría estando conectado a la aplicación, lo cual desprestigiaría el propósito de las notificaciones. Por eso se implementará una tarea recurrente, que se ejecuta con cierta periodicidad actualizando los feeds de todos los usuarios.
En Rails existen diversas maneras de implementar esto, así como gemas para ejecutar tareas de este tipo. Se opta por utilizar una gema conocida como Sidekiq (ver 4.1.1.4), que se utiliza para resolver todo tipo de tarea en Background, con el fin de aliviar la carga sobre el servidor web (el proceso que ejecuta la aplicación). El beneficio de esto es que la aplicación podrá seguir

atendiendo peticiones entrantes, mientras que se actualiza la base de datos a través de otro proceso independiente.
Los pasos a seguir son:

- Agregar la gema en el Gemfile: `gem 'sidekiq'.`
- Sidekiq se debe correr como proceso aparte y además utiliza Redis (un sistema de almacenamiento de clave-valor) para poder registrar los procesos planificados. Por eso se agregan las siguientes líneas a Procfile.dev (archivo que arranca los procesos de la aplicación):

```
redis:   redis-server
sidekiq: bundle exec sidekiq
```

- Sidekiq se ocupa únicamente de planificar tareas cuando se lo ordena y de ejecutarlas en el momento debido. Existe otra herramienta, conocida como Sidetiq, que permite registrar tareas que se cumplen con periodicidad fija, es decir que no dependen de un suceso (u orden) que puede variar en el tiempo (para más detalles ver 4.1.1.4). Se agrega Sidetiq de la siguiente manera: `gem 'sidetiq'`
- Sidekiq cuenta con una interfaz de usuario para gestionar las tareas que se corren en background. Sidetiq extiende dicha interfaz para las tareas recurrentes. Para tener acceso a esta extensión se agrega lo siguiente en routes.rb:

```
require 'sidetiq/web'
require 'sidekiq/web'
mount Sidekiq::Web => '/sidekiq'
```

- Por último se corre `bundle install` y se reinicia el servidor.

Lo único que falta es agregar el Worker que debe ser ejecutado por Sidekiq, con el fin de actualizar periódicamente las entradas de todos los feeds. Un worker es una clase que contiene código que se procesa en background mediante Sidekiq/Sidetiq.

Se decide arbitrariamente realizar la actualización cada 15 minutos, dado que es suficientemente seguido para mantener actualizado al usuario, sin sacrificar demasiada capacidad de procesamiento. Habrá que tener en cuenta que la aplicación eventualmente deberá procesar miles de feeds, cada uno de estos con alrededor de 50 entradas cada vez que se actualice. Por lo tanto habrá que tener cuidado de no agotar la memoria y capacidad de procesamiento en un futuro. Para prevenir lo último, vamos a incorporar procesamiento en batches a nuestro FeedWorker para limitar la ocupación de memoria a un máximo de 100 instancias de Feed por vez. El código correspondiente se puede observar en `app/workers/feed_worker.rb`.

5.4.1.3.2. Marcar entrada como leída/no leída

Para poder implementar esta funcionalidad, ante todo se tendrá que agregar un campo al modelo Entry, que nos indique si una entrada fué leída o no. Luego existen dos opciones:

1. Crear una acción nueva en el EntriesController que se ocupe únicamente del cambio de estado de leída a no leída y viceversa. Esto agrandaría al controlador en forma innecesaria, considerando que objectivamente sólo se requiere modificar un atributo de una entrada. Para esos casos se cuenta con una acción estándar llamada *update*.
2. Aprovechar la acción *update* para actualizar el atributo `read` de la instancia de Entry correspondiente.

A raíz de lo anterior queda evidente cual es la mejor solución en este caso. Se realizará un llamado a través de un link a la acción *update* del EntryController, con el fin de marcar o desmarcar como leída cada una de las entradas. La respuesta se llevará a cabo mediante AJAX con el fin de no tener que volver a descargar la página completa, sino simplemente poder actualizar la entrada particular dentro de la página cargada. Esto último se tendrá que implementar con JavaScript, dado que desde el servidor no se puede ejecutar código en el navegador. Ver sección 4.3 para más información sobre JavaScript.

En resumen se seguirán los siguientes pasos:

1. Crear la acción (método) `update` en el EntryController.
2. Crear un método `update` en EntriesService para prevenir que el controlador quede sobrecargado en un futuro. Es una buena práctica mantener los controladores con la menor cantidad de código posible.
3. Para poder modificar el atributo `read` del modelo Entry, se tendrá que agregarlo al listado de atributos accesibles (mediante el método `attr_accessible`). Sino ActiveRecord (ver 4.1.3.2) no permitirá actualizar dicho campo. Esto se conoce como *whitelisting*, y constituye una medida de seguridad que requiere explicitar el acceso a los datos.
4. Configurar la ruta a la acción `update` (en routes.rb) en caso que todavía no exista.
5. Implementar el link para marcar la entrada. El link va a solicitar la respuesta en formato JS (JavaScript).
6. Escribir el código de JavaScript que contenga los datos del servidor y modifique el estado de la entrada (ver código en app/views/entries/update.js.erb).

Adicionalmente para reflejar los cambios visualmente se tendrá que:

1. Crear estilos en main.css para la clase `read` y `unread`.
2. Agregar la lógica necesaria a la vista de entradas para que pueda asignar las clases según el estado de cada entrada. Esto se logra por medio de un *helper*. Se trata

simplemente de un método que encapsula la lógica necesaria para visualizar correctamente una vista. De esta forma se separa el código de la vista.

5.4.1.3.3. Marcar entrada como favorita/no favorita

Para implementar esta característica se utiliza ampliamente el código generado para marcar una entrada como leída/no leída. De hecho lo único que se tendrá que agregar es:

1. Un nuevo link para favoritos, agregando un helper que lo genere automáticamente con los parámetros requeridos.
2. Agregar el código JS para manejar los cambios de HTML a través de AJAX.
3. Agregar los estilos CSS que permiten modificar visualmente el ícono de la estrella en el link de marcar favorita.

Con esto se puede marcar y desmarcar entradas. Se puede concluir que, gracias a la forma genérica de la que se implementa el marcado de entradas leídas/no leídas, se ha podido reutilizar gran parte de código para el marcado de favoritas.

5.4.1.3.4. Barrido diario de entradas

Hasta ahora se implementó la actualización de las entradas de forma automática y continua con un intervalo de 15 minutos. Esto sirve para mantener entradas actuales en todo momento. En un entorno de desarrollo, esto probablemente no signifique un consumo excesivo de memoria en la base de datos. Sin embargo, se debe tener en cuenta que más adelante (en producción) esto podría saturar la capacidad del servidor de base de datos en poco tiempo, considerando una gran cantidad de usuarios.

Dicho esto, simplemente se tendría que implementar un mecanismo para barrer la base de datos, dejando una cantidad arbitraria de entradas por feed. Para esto se decidió limitar la cantidad de entradas a 50 luego de cada barrido, siendo las más actuales las que permanecen guardadas. Además sólo se eliminarán los feeds que no son favoritos, de manera que los favoritos serán guardados permanentemente y sin ser considerados dentro del límite de entradas.

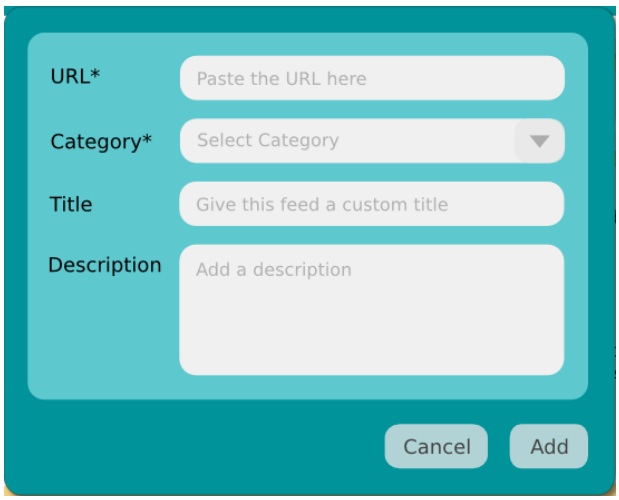
Se aclara que con “cantidad límite de entradas”, no se quiere expresar que jamás podrá haber más de dicha cantidad de entradas por feed. En cambio se refiere a que luego de un barrido, sólo permanecerán en la base de datos las 50 entradas más actuales de cada feed, considerándose en esta cuenta sólo a las entradas no favoritas. Como ya se remarcó, las favoritas permanecen siempre.

Arbitrariamente se establece el barrido para ser ejecutado en background mediante Sidekiq/Sidetiq (ver 4.1.1.4) una vez por día a las 00:01 hs. Se hará en ese horario porque se estima que habrá menos consultas. Considerando que eventualmente accederán usuarios de

todo el mundo con diferentes horarios, probablemente se tendrá que mejorar esta estrategia en un futuro. El intervalo se establece en 24 horas porque se sostiene que es preferible preservar la capacidad de procesamiento antes que liberar memoria. De vuelta, dependiendo del tráfico de la aplicación en producción, eventualmente se tenga que reconsiderar esta gestión. Adicionalmente, se decide realizar esta tarea un minuto luego de la última actualización de las entradas, para prevenir conflictos entre ambos procesos en background y además lograr que se eliminen las últimas entradas descargadas.
Para leer la implementación del worker correspondiente, ver el archivo `app/workers/entry_worker.rb`.

5.4.1.3.5. Agregar feed

Para agregar feeds se tendrá un botón con el símbolo '+' en la parte superior izquierda de la vista Home. Al presionarlo, la aplicación responderá con una ventana emergente, pidiendo los datos correspondientes del feed a agregar. El aspecto de dicha ventana es el siguiente:



El formulario se puede cerrar mediante el botón 'Cancelar'. Para agregar el feed se debe haber ingresado/seleccionado los datos mínimos requeridos que son URL y Categoría (marcados debidamente con un asterisco). Por defecto la categoría será 'Indefinida'. El formulario se envía y valida del lado del servidor, mediante el botón 'Agregar'. En caso de que los datos sean correctos, se cierra la ventana y aparece un mensaje que notifica que la operación ha sido exitosa. En caso contrario se marcarán con rojo los campos que contengan errores y además aparecerá un mensaje describiendo el error. El usuario podrá corregir los campos necesarios y volver a enviar la información o cancelar la operación.

La ventana emergente se implementa con un componente, conocido como *modal*, de **Bootstrap** (ver 4.5.3.1). Para implementar esto, simplemente se copia la estructura HTML y los estilos CSS descritos en la página de referencia de Bootstrap.
En Rails existe un conjunto estándar de *helpers* para crear formularios, sin embargo actualmente se tiende a usar una de 2 gemas: Formtastic y Simple Form. En este caso, se decide utilizar Simple Form (ver 4.1.1.6), por previa familiarización.
Al seleccionar el signo '+' para agregar feeds, la aplicación creará y enviará un formulario dentro de un modal y luego lo mostrará mediante JavaScript. El usuario podrá llenar dicho formulario y enviarlo para crear el feed nuevo presionando el botón de 'Agregar'. En caso que quiera cancelar la operación, simplemente tendrá que presionar el botón de 'Cancelar' y desaparecerá la ventana.
Durante la implementación de esta ventana se deben considerar **2 puntos clave**:

1. La validación de la Url: Asegurar que la dirección sea válida para un feed y en caso contrario notificar el error.
2. Restringir las categorías: Se deben poder seleccionar únicamente las categorías asociadas al usuario que se encuentra conectado. Hasta el momento se mostrarían absolutamente todas las categorías.

5.4.1.3.5.1. Validar URL

La primer cuestión se soluciona utilizando una gema específica para poder validar la url. Los pasos a seguir son:

1. Crear una carpeta `validators/` bajo el directorio `app/`.
2. En dicha carpeta, crear un validador con el nombre `FeedUrlValidator`. Esto se hace para evitar la saturación del modelo con código que no es específico del mismo. Si bien probablemente no se volverá a utilizar el validador en otra parte de la aplicación, es una buena práctica crear un validador aparte, para eventualmente poder reutilizarlo en otro modelo o entidad en general. El código correspondiente se encuentra en `app/validators/feed_url_validator.rb`.
3. Agregar la validación en el modelo Feed a través de la siguiente línea de código: `validates :url, feed_url: true`.

A partir de este momento la URL se valida a través del modelo, con lo cual ya no habrá que preocuparse por feeds inexistentes o defectuosos a la hora de ser agregados.

5.4.1.3.5.2. Restringir categorías y otras entidades

Con la iniciativa de restringir las categorías, se tomará un paso más allá y generalizar la perspectiva del problema.

La cuestión es que la aplicación alberga usuarios que tienen asociados: categorías, feeds, notificaciones, filtros y entradas. La asociación directa es entre usuarios y categorías, mientras que las demás entidades están asociadas a un usuario por medio de la relación transitiva desde cada categoría a las dichas entidades (ver DER en 5.2).

Por ejemplo, para recuperar los feeds de un usuario se buscan primero todas las categorías del usuario y luego se buscan los feeds correspondientes a cada una de esas categorías. Para llegar a los filtros, notificaciones y entradas, se tiene que partir de los feeds recuperados anteriormente, que se asocian con estos.

Esta red de asociación asegura mantener los datos separados entre los diferentes usuarios. Sin embargo requiere que las entidades se restrinjan en forma explícita según el usuario, cada vez que se quiera recuperar los datos. Es por eso que, a la altura actual del desarrollo, en la selección de categorías para agregar un feed, aparecerán todas las categorías para ser seleccionadas (de todos los usuarios). Esto se puede restringir en forma explícita, recuperando primero el usuario y buscando las categorías a través de este. El problema es que no será el único lugar donde se tendrá que hacer esto, lo cual significa tener que acordarse y escribir el mismo código, cada vez que se tenga que separar los datos.

Si se decide tomar este camino, de separar los datos manualmente en todo momento, no sólo se tendrá que escribir mucho código (mayormente repetitivo), sino que además la aplicación será propensa a fallas, dada la probabilidad de olvidar restringir los datos. Además la navegación desde el usuario hasta las entidades que se requieren restringir, significa un proceso ineficiente, considerando la necesidad de repetidas consultas a diferentes tablas de la base de datos.

Por suerte existe una solución mucho más simple que permite lo que se denomina ‘multi-tenancy’, que es la separación de los datos de varios usuarios dentro de una misma base de datos. Existe una gema (Acts As Tenant, ver 4.1.1.8) que permite separar los datos por usuario en forma transparente y eficiente.

Esta funcionalidad se aplica de la siguiente manera:

1. Agregar `gem 'acts_as_tenant'` en el Gemfile.
2. Correr `bundle install`.
3. Agregar los filtros apropiados en `app/controllers/application_controller.rb`:

   ```
   set_current_tenant_through_filter

   before_filter :find_current_tenant
   ```

```
def find_current_tenant
  set_current_tenant(current_user)
end
```

4. Agregar el campo `user_id` como tipo `integer` en cada uno de los modelos que sean específicos de cada usuario. Estos son: Filter, Notification, Category, Feed y Entry.
5. Agregar la declaración `acts_as_tenant :user` en cada uno de los modelos anteriores. Esto le indica a la gema el campo que debe usar de criterio para resolver la tenencia.
6. Crear y correr las migraciones mediante `rails g hobo:migration`.

A partir de este momento todas las entidades están restringidas a sus respectivos usuarios. El filtrado es más eficiente que el enfoque propuesto anteriormente (sin usar la gema), debido a que cada modelo tiene guardado una referencia directa (`user_id`) al usuario correspondiente. Esto permite, por ejemplo, obtener todos los feeds de un usuario mediante una única consulta, sin tener que recurrir previamente a las categorías del usuario como intermediarias.

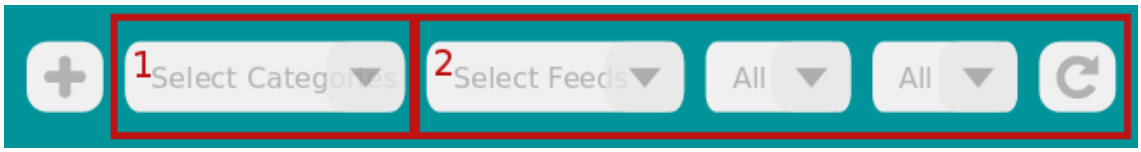
5.4.2. Iteración 2

5.4.2.1. Filtrado de entradas en Home

Con filtrar entradas se hace referencia a la funcionalidad para poder visualizar las entradas, según categorías, feeds y los estados (leído/favorito). Es decir, se tendrá un conjunto de campos para elegir cada uno de estos atributos, y utilizarlos de criterios de selección.
Para las categorías y feeds la selección será múltiple, lo cual significa que se podrá elegir varias categorías y feeds correspondientes al mismo tiempo. En cambio, los estados (leído/favorito) serán de selección simple, incluyendo el estado 'Todos', mediante el cual no se discriminará entradas bajo el criterio correspondiente.
Una característica importante a tener en cuenta es que, cuando se seleccionan una o varias categorías, los feeds que se pueden seleccionar a continuación deben estar relacionados con alguna de las categorías elegidas. Por lo tanto, se implementará un mecanismo para forzar el filtrado de los feeds seleccionables, en el momento en que se elija las categorías.
Para esto último se contará con 2 formularios:

1. **Formulario de selección de categorías**: Se envía automáticamente cuando se selecciona la categoría. El servidor responde refrescando la página con los feeds correspondientes en la lista de selección.
2. **Formulario de entradas**: Este formulario contiene la selección de feeds, leída/no leída y favorita/no favorita. A diferencia del formulario anterior éste contará con un botón para enviar los datos, con el fin de evitar que se envíe información continuamente.

En la siguiente imagen se muestran los formularios según el diseño.



Una dificultad adicional es el envío de las categorías con el formulario de entradas, dado que se debe mantener esta información al aplicar el filtro. La solución es agregar un campo oculto al formulario de entradas que contiene las categorías.

Para el formulario de entradas se utilizará la misma gema (Simple Form, ver 4.1.1.6), que aquella para implementar el formulario de 'Agregar Feed'. El formulario de selección de categorías, sin embargo, se implementará como formulario estándar, es decir con los helpers que provee Rails, porque sólo se enviará información de la categoría, que no estará asociada de forma directa al modelo Entry.

Para lograr que el formulario de selección de categorías se envíe automáticamente, una vez realizada la selección, se tendrá que escribir código JavaScript que envíe el formulario, al dispararse el evento de 'change' sobre el campo de selección.

El filtrado de las entradas propiamente dicho se podría implementar manualmente, pero existen soluciones más flexibles que ya se encuentran implementadas. Las principales opciones son MetaSearch y Ransack. En este caso se optó por la última, por conocer su sintaxis y funcionamiento de antemano. Ransack (ver 4.1.1.9) permite implementar un formulario de búsqueda con mínimas dificultades.
Los pasos a seguir son:

1. Agregar al Gemfile:
    a. `gem 'polyamorous'` (Ransack depende de esta gema)
    b. `gem 'ransack'`
2. Correr `bundle install`.
3. Crear el formulario en `_entries_search_form.html.haml` (en app/views/entries/) mediante el helper de Ransack `search_form_for`. Se puede utilizar la sintaxis de Simple Form al agregar la opción `builder: SimpleForm::FormBuilder` al helper.
4. Agregar la lógica adicional en el controlador `EntriesController`. La parte de Ransack es la siguiente:
   `search = Entry.ransack(search_params)` (se pasan los parámetro del formulario)
   `entries = search.result(distinct: true)` (se obtienen los resultados distintos para evitar repetidos)

Con lo anterior ya se puede realizar las búsquedas. Para la selección múltiple habrá que agregar la opción correspondiente a los campos de categorías y feeds. Además se agrega la opción de buscar por "Todos", para el caso en que el usuario quiera omitir el filtrado por determinados campos.
Dado que el filtrado de entradas se encuentra funcionando por completo, sólo falta ajustar los estilos para que queden acorde con el diseño. Estos detalles se omiten por no aportar a la implementación funcional de la aplicación. Sin embargo, vale aclarar que se agrega un plugin de JavaScript (Select2) con el fin de mejorar la manera en que se realiza la selección múltiple de categorías, feeds y estados. Una de las características que agrega este plugin es la búsqueda opcional sobre los campos de selección. Esto facilita considerablemente el manejo del filtro, especialmente cuando el usuario cuente con grandes cantidades de categorías y feeds al mismo tiempo.
Con esto se dá por terminada la vista de Home.

5.4.2.2. Gestión de Categorías

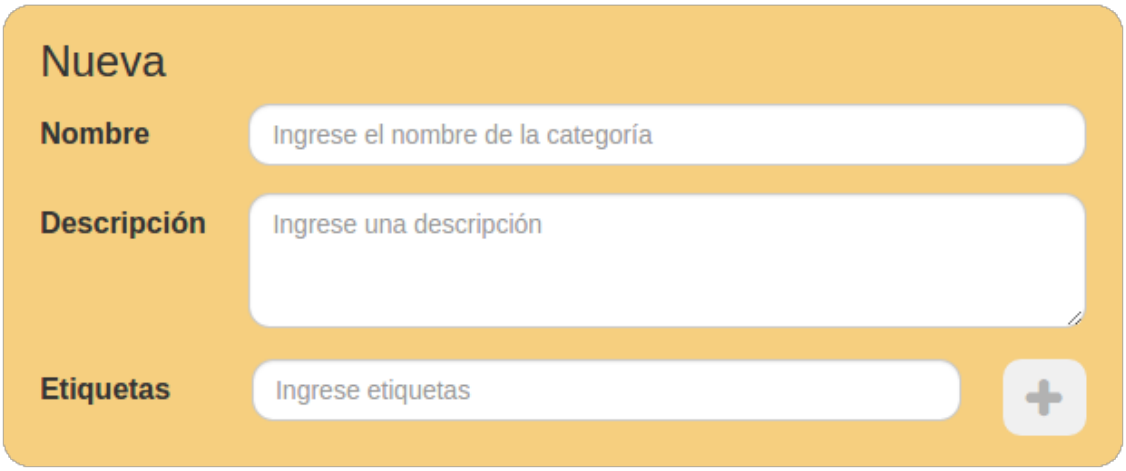
La Gestión de Categorías permite visualizar, crear, modificar y eliminar categorías. Además se pueden desplegar los feeds asignados a cada categoría, con el fin de visualizar la información correspondiente, y eliminarlos.
Estructuralmente la vista consta de 2 partes:

1. **Formulario de Categorías**: Permite crear categorías nuevas.
2. **Listado de Categorías**: Muestra todas las categorías en orden alfabético ascendente.

A continuación se desarrollan las diferentes funcionalidades de la gestión de categorías.

5.4.2.2.1. Crear Categoría

Para poder crear una categoría se implementa un formulario con la gema Simple Form (ver 4.1.1.6) mencionada anteriormente. El formulario cuenta con los campo `name`, `description` y `tags`. El campo `editable` del modelo Category sólo se utiliza para la categoría 'Indefinida', con el fin de asegurar que esta no se pueda eliminar ni modificar.
El formulario se envía mediante AJAX, con lo cual la respuesta será mediante .js.erb (`refresh_all.js.erb` en app/views/categories/). Esto significa que en vez de volver a cargar la página completa, se ejecuta código JavaScript enviado desde el servidor web. El código recarga ambas partes de la vista de categorías. Estas son: El formulario, con el fin de limpiar los campos recién enviados, y la lista de categorías para que esta pueda mostrar todas las categorías, incluyendo la nueva.
Para que el cambio no resulte visualmente abrupto, el código JS (específicamente JQuery) se encarga de hacer desaparecer lentamente el contenido obsoleto, luego carga los datos en forma invisible y finalmente hace reaparecer el contenido nuevo en forma lenta. Esto último simplemente sirve para hacer la transición más agradable para el usuario.
El formulario resultante se muestra en la siguiente captura de pantalla.



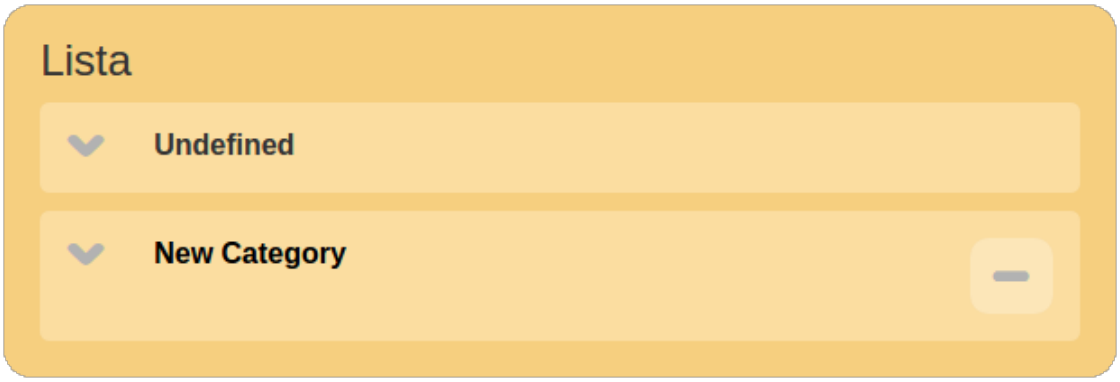
5.4.2.2.2. Ver Categorías

Las categorías se pueden ver en el listado que se encuentra por debajo del formulario. Del lado derecho de cada una de las categorías, aparece un botón con el símbolo "-" que sirve para eliminar la categoría. Este botón está implementado con un link a la acción `destroy` del controlador de categorías. Al presionarlo, se envía una petición mediante AJAX, con lo cual una vez eliminada la categoría, se actualiza únicamente el listado de categorías evitando tener que refrescar toda la página. De vuelta esto se realiza con js.erb (en app/views/categories/refresh_list.js.erb), mencionado anteriormente.
A la izquierda de algunas categorías aparece una flecha que permite desplegar los feeds de cada categoría. En caso de no contar con esta flecha, significa que la categoría no tiene feeds asociados. el despliegue se implementa con un componente de Bootstrap conocido como **Collapse**. Esto también se utilizó para lograr el comportamiento de contracción de las entradas en la vista Home. Para implementar esta funcionalidad se debe agregar la clase `'collapse'` al contenedor (generalmente `div`) que se desea contraer, y agregar los atributos `data-toggle='collapse'` y `href` con una referencia al contenedor a contraer.

Finalmente si se pasa el cursor por encima del nombre de la categoría, aparece un panel pequeño mostrando la descripción y las etiquetas en caso de estar presentes. Nuevamente esto se logra con un componente de Bootstrap llamado **Tooltip**. Estos componentes se implementan aplicando los atributos `data-toggle="tooltip" data-placement="top" title="Descripción y etiquetas"` sobre el elemento que contiene el nombre de la categoría.

- `data-placement="top"` indica que la información se mostrará por encima del nombre seleccionado.
- `title` contiene el texto a mostrar.

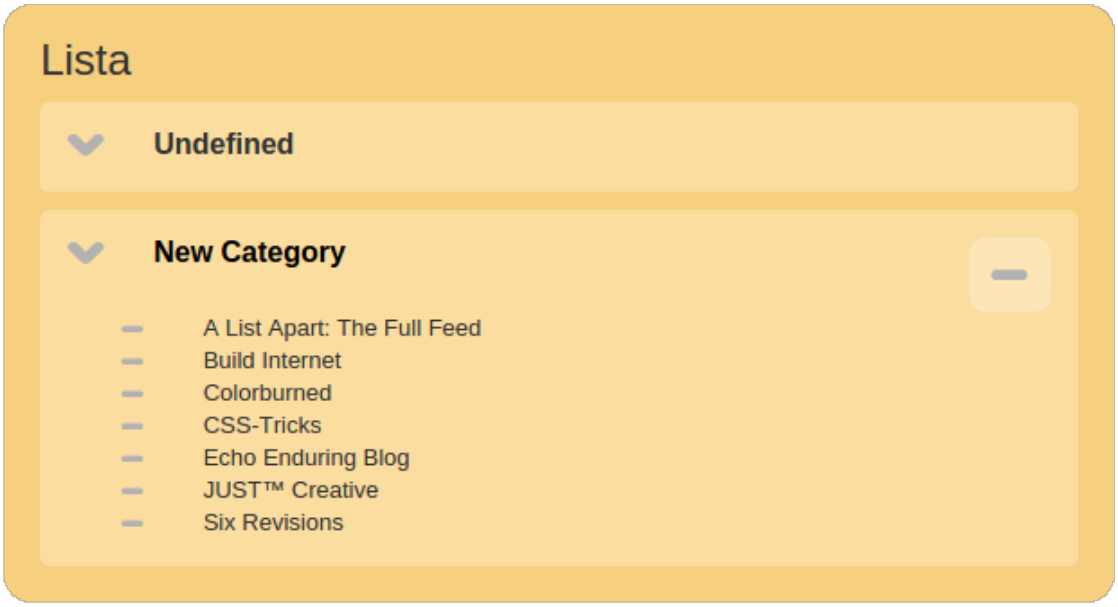
Para activar los tooltips se requiere un paso adicional que consiste en ejecutar código JS sobre todos los elementos que tienen tooltips asociados. La manera más simple (ilustrada en la referencia de Bootstrap) es ejecutando: `$('[data-toggle="tooltip"]').tooltip()`.
Con esto se encuentra a todos los elementos con tooltip a través de JQuery (ver 4.3.1) y luego se ejecuta el método `tooltip` de Bootstrap.
Abajo de muestra una captura de pantalla con el listado de categorías resultante.



Se puede ver la categoría indefinida (“Undefined”) así como una categoría creada por el usuario. Ambas categorías contienen feeds, dado que figura una flecha a la izquierda de cada una. Además, se puede ver que la categoría indefinida no cuenta con el botón para borrarla.

5.4.2.2.3. Ver Feeds por Categoría

Hasta el momento, en la lista de categorías se pueden visualizar todas las categorías. Se quiere lograr que a la izquierda de cada categoría que tenga feeds asociados aparezca una flecha, y que al ser seleccionada despliegue todos los feeds correspondientes. A su vez, se debe tener la posibilidad de 'cerrar' la categoría, para volver a esconder los feeds.
Para esto se utilizará el mismo componente de Bootstrap (ver 4.5.3.1) empleado para expandir y contraer las entradas de los feeds, en la vista de Home. La diferencia con la implementación anterior del componente *Bootstrap Collapse*, es que en este caso se podrá tener desplegadas varias categorías a la vez. En el caso de las entradas, al abrir una, se cerraba la anterior, si hubiera alguna abierta. Esto resulta útil para las entradas dado que son muchas, para mejorar la visibilidad de la información y facilitar el manejo. En el caso de las categorías, al no ser tantas, resulta más cómodo, desde el punto de vista del usuario, dejar las categorías abiertas.
A continuación se muestra una captura de pantalla de la lista de categorías.



Se puede observar 2 categorías: La primera “Undefined” se encuentra colapsada, mientras que la segunda “New Category” se encuentra expandida. Bajo esta última se muestra un listado de feeds.

5.4.2.2.4. Modificar Categoría

Las categorías que se encuentran en la lista deben poder modificarse. Para esto se sostiene que la forma más intuitiva es seleccionando el nombre de la categoría desde la lista, con lo cual se cargan los datos correspondientes en el formulario en la parte superior de la vista. Desde el formulario se pueden realizar las modificaciones y finalmente guardar los cambios.
Al utilizar el mismo formulario tanto para la creación como la modificación de categorías, es necesario incorporar un botón que nos permita volver desde la edición de una categoría a la creación de una nueva categoría. Una forma de lograr esto es seleccionando la vista de categorías en el menú principal. Sin embargo, no se debería tener que volver a entrar a la sección de categorías para poder realizar esta operación. Por eso se incorpora un botón condicional “Nueva” al formulario, que sólo se muestra cuando se está editando una categoría existente. Dicho botón permite limpiar el formulario para crear una categoría nueva. A continuación se muestra el formulario de categorías en estado de edición. Obsérvese el botón “Nueva” en el marco superior derecho.

Un feed jamás debe existir sin estar asociado a una categoría. Debido a esta restricción se creó la categoría “indefinida” (o “undefined”), para cubrir el caso en que el usuario no tenga categorías creadas e intente agregar feeds de todas formas. El usuario no debe poder editar ni borrar la categoría indefinida, por lo cual se tendrá que habilitar la opción de editar y borrar para todas las demás categorías, salvo la indefinida. Esto se realiza por medio del campo *editable* de Category, y se asegura a través de un Callback en el modelo Category.
Un Callback es un método que se llama cuando se cumple una condición o evento. En el caso actual se tendrá que implementar un Callback para 2 casos: Modificación y Borrado. Es decir, el usuario no debe poder modificar ni borrar la categoría 'indefinida', por lo cual se tendrá que llamar un Callback frente a 2 eventos diferentes. Esto se implementa como sigue en app/models/category.rb

```
before_update :check_if_editable

def check_if_editable
  raise I18n.t('errors.can_not_be_edited') unless editable
end
```

Se podría utilizar el mismo método y asociarlo a los 2 eventos (*update* y *delete*), sin embargo se decide crear 2 métodos para que el mensaje de error sea más preciso.
Con esto el usuario se verá incapaz de cambiar o eliminar la categoría indefinida. Esto sucede a nivel del modelo, con lo cual se puede estar seguro de que no se podrá eludir esta restricción al menos desde el cliente.

Aún así sería conveniente esconder la opción de edición y borrado del lado del cliente, es decir que ni siquiera aparezcan los botones correspondientes, de manera que se evita realizar validaciones en el servidor. Esto se logra con una simple condición, haciendo que el vínculo de edición sólo se muestre cuando el valor de *editable* de una categoría sea verdadero. Algo muy similar se hará para el borrado de la categoría.

5.4.2.2.5. Eliminar Categoría

Debe ser posible eliminar una categoría entera con sus feeds correspondientes. Para asegurar que la categoría indefinida no se borre, se utiliza del lado del servidor el Callback correspondiente en app/models/category.rb como se mostró para la edición anteriormente:

```
before_destroy :check_if_destroyable

def check_if_destroyable
  raise I18n.t('errors.can_not_be_destroyed') unless editable
end
```

Al igual que en el caso de la edición, también sería favorable esconder la opción de edición para la categoría indefinida. Por eso el botón de borrado no aparece para la categoría indefinida.

5.4.2.2.6. Eliminar Feed

Cuando una categoría se encuentra desplegada, se muestran los feeds asociados por debajo. Se quiere lograr que cada feed tenga al lado izquierdo un símbolo '-' que sirva de botón para eliminar el feed. Esto se implementa de manera muy simple, dado que el controlador de feeds ya existe, así como sus rutas y sólo implica implementar el método `destroy`, la respuesta por AJAX y el botón en sí.

Se agrega, por lo tanto, el vínculo para borrar el feed al HTML (o mejor dicho HAML):

```
= link_to feed_path(feed), method: :delete, remote: true do
  .icon.small
    e
```

Al estar utilizando una fuente como origen para los íconos (ver 4.5.2), se debe referir a cada ícono con una letra y encapsular dicha letra en un elemento que tenga definido el estilo de fuente que contiene los íconos. Es así como la letra “e” se traduce en un símbolo ‘-’.

Si se selecciona el ícono se borra el feed y se muestran los cambios refrescando el listado de categorías. Como ocurre con la mayoría de las peticiones de nuestra aplicación, esto se lleva a cabo mediante AJAX. Por lo tanto no se refresca la página completa, sino sólo las partes necesarias para reflejar los cambios realizados.

Si se vuelve a observar la imagen de la sección 5.4.2.2.3, se puede ver que cada feed de la categoría “New Category” lleva un botón con el símbolo “-” adyacente.

5.4.3. Iteración 3

5.4.3.1. Gestión de filtros

Los filtros se utilizan principalmente para excluir ciertas entradas, acorde con palabras clave asociadas a cada feed individualmente. Es decir, cada feed puede tener un filtro que establece un conjunto de palabras clave. Si al actualizar las entradas de un feed apareciera alguna de las palabras clave del filtro asociado, la entrada correspondiente será descartada o almacenada, dependiendo del tipo de filtro que se establece.
Los 2 tipos de filtro y sus comportamientos son los siguientes:

1. Lista negra: Se descartan todas las entradas que *contienen* al menos una de las palabras clave.
2. Lista blanca: Se descartan todas las entradas que *no contienen* al menos una de las palabras clave.

Más allá de servir para excluir las entradas con el fin de no mostrarlas en la vista Home, los filtros también se tendrán en cuenta para las notificaciones. Se realizará el filtrado antes de pasar por el gestor de notificaciones, con lo cual el filtro prevendrá que se notifique al usuario de temas que luego no aparecerán en su vista de Home.

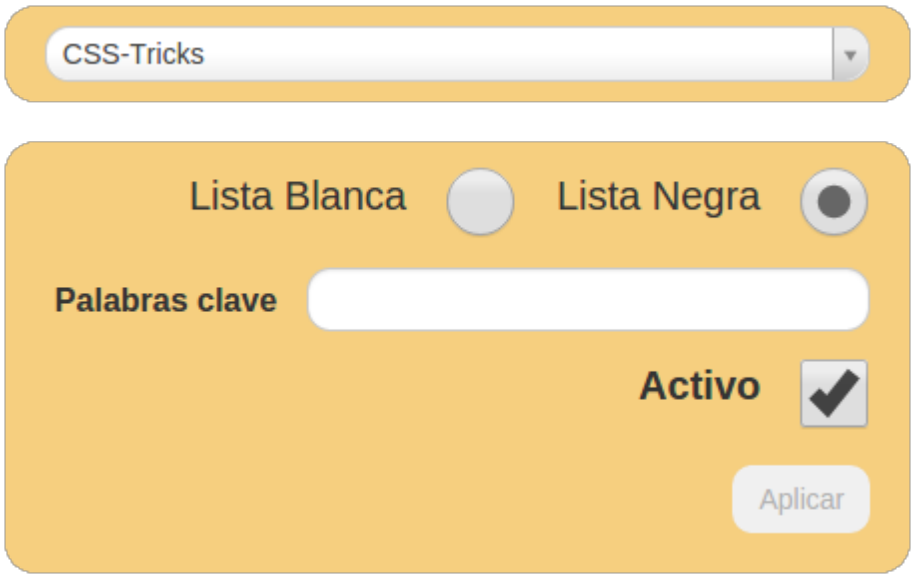
Para implementar la vista de filtros se comienza por crear el controlador correspondiente:
`rails g controller filters`
Una vez hecho esto, se agregan las rutas a routes.rb: `resources :filters`.
Luego se debe crear la acción `index` en el archivo `filters_controller.rb`, que se acaba de generar en app/controllers.
Finalmente se crea la vista de index agregando el archivo `index.html.haml` en la carpeta app/views/filters.
Para poder visualizar la vista por ahora se tendría que acceder directamente a través de la URL `http://localhost:3000/filters`. Esto se puede mejorar rápidamente accediendo al partial `_aside.html.haml` que forma parte del layout, y agregando el vínculo anterior mediante un helper generado automáticamente por Rails.
Con esto se concluye la base para poder gestionar los filtros.

5.4.3.1.1. Alta de filtros

Con el fin de establecer un filtro nuevo, ante todo, se requiere seleccionar el feed correspondiente en el campo de selección de la vista de filtros.

Una vez seleccionado el feed se desplegará el formulario para crear un filtro nuevo o (en caso que ya exista un filtro para dicho feed) modificar el filtro correspondiente. En la siguiente imagen se muestra el campo para seleccionar el feed, antes de desplegar el formulario.

Para cada feed se podrá tener un sólo filtro. Como se mencionó anteriormente, éste tendrá que ser de un único tipo y contar con una lista de palabras clave. Además cada filtro cuenta con un campo que indica si el filtro se encuentra activado o no.
El formulario de filtros se construye con Simple Form (ver 4.1.1.6). El detalle de la implementación de dicho formulario se omite, dado que no constituye una característica que no se haya mencionado anteriormente. El formulario, visible una vez seleccionado un feed, es el siguiente:



Un componente distintivo de la vista de filtros es el uso de etiquetas, para mejorar el manejo de las palabras clave. Esto significa que, en vez de mostrar las palabras clave en su campo como texto plano separadas por comas, se visualizará cada palabra como una etiqueta con una cruz al lado, para poder eliminar la etiqueta individualmente. Esto no aporta a la funcionalidad de la aplicación, porque no altera la representación interna del listado de palabras. Sin embargo facilita el manejo de las palabras clave. Para lograr lo anterior se utiliza un plugin de Bootstrap a través de una gema para Rails, conocida como *bootstrap-tagsinput-rails*.
Los pasos para adoptar la gema al proyecto son:

1. Agregar la gema al Gemfile: `gem 'bootstrap-tagsinput-rails'`
2. Instalar la gema a través de la consola: `bundle install`
3. Al tratarse de un plugin de javaScript (ver 4.3) se tiene que agregar la siguiente línea a app/assets/application.js: `//= require bootstrap-tagsinput`

4. El plugin define un conjunto de estilos, con lo cual se debe agregar la siguiente línea a app/assets/application.css: `*= require bootstrap-tagsinput`
5. Finalmente sólo queda agregar un atributo data al input de palabras clave:

```
= f.input :keywords, input_html: {data: {role:'tagsinput'}}
```

Con esto el plugin se aplicará al cargar la página. Debido a que este plugin sólo se aplica automáticamente luego de una carga completa de la página, se presenta un pequeño inconveniente a la hora de actualizar contenido con AJAX. Cuando se refresca el formulario remotamente (es decir, mediante AJAX), luego de guardar el filtro, el campo de palabras clave no toma las etiquetas porque no se vuelve a correr el código JS de la carga de página. Por lo tanto la solución es incluir un llamado explícito al plugin con la respuesta de AJAX. Con esto se consigue actualizar el formulario e inmediatamente después aplicar el plugin sobre el campo de palabras clave.

5.4.3.1.2. Baja de filtros

Para borrar un filtro se agrega un botón de 'eliminar' al formulario de filtros. Dicho botón sólo aparece cuando el filtro se encuentra guardado en el sistema. Por lo tanto, cuando no aparece se puede inferir que el filtro no se encuentra aún en la base de datos del sistema.
Al eliminar el filtro se recarga el formulario completo, asumiendo que el usuario no tendrá interés en crear un nuevo filtro para el mismo feed. La lista de selección queda entonces en su posición inicial.

5.4.3.1.3. Modificación de filtros

La modificación de filtros no requiere mayores cambios en base a los componentes desarrollados anteriormente.
Se cuenta con la selección de feeds y el formulario de filtros, además de las respuestas generadas por el servidor cuando se crea un filtro. Estos componentes se pueden reutilizar como se ha hecho análogamente en la sección de categorías. Lo único que falta, por lo tanto, es modificar ligeramente la acción de *index* y crear la acción *update* en app/controllers/filters_controller.rb.
Hasta el momento, cada vez que se seleccionó un feed del campo de selección, se cargó el feed correspondiente de la base de datos y se le asignó un filtro nuevo. Ahora se tendrá que tener en cuenta el caso en que se desee modificar un filtro preexistente. Con este propósito se agrega la siguiente condición a la acción *index*:

```
if @feed.filter
  @filter = @feed.filter
else
  @filter = Filter.new feed_id: @feed.id
end
```

El código anterior carga el filtro asociado al feed en caso que ya exista. En caso contrario crea un filtro nuevo y lo asocia al feed. El helper del formulario se encarga de establecer la ruta correcta, la cual puede apuntar tanto a la acción *create* cómo *update* dependiendo de la naturaleza del filtro.
Finalmente se define la acción *update* en el controlador de filtros y se adopta la misma respuesta de AJAX utilizada para la acción *create*.

5.4.3.2. Filtrado

El filtrado debe ser considerado a la hora de actualizar los feeds, es decir al descargar nuevas entradas. No se aplica el filtrado a entradas preexistentes en el momento en que se crea un filtro nuevo.
Cuando se actualizan los feeds, se llama al método `update_entries` del modelo Feed.
Hasta el momento dicho método se ocupaba de obtener el feed mediante la URL correspondiente y luego llamaba al método `add_entries`, para persistir a cada una de las entradas en la base de datos. Esto se seguirá llevando a cabo, con la diferencia de que se implementará un método intermedio que, una vez obtenido el feed, revisa el contenido de cada una de las entradas, descartando aquellas que no pasen el filtro.
El método `update_entries` queda de la siguiente manera:

```
def update_entries
  feed = Feedjira::Feed.fetch_and_parse(url)
  raise 'Feed format issue' if feed.is_a? Numeric
  unless self.etag == feed.etag && self.last_modified ==
feed.last_modified
    apply_filter!(feed) if filter._?.active
    self.etag = feed.etag
    self.last_modified = feed.last_modified
    Entry.add_entries(feed.entries, self.id, user_id)
  end
```

```
  feed
end
```

En **verde y negrita** figura el código agregado para hacer funcionar el filtro. Como se puede ver, el método de `apply_filter!` se ejecuta solamente cuando existe un filtro asociado y activo. Se puede observar que el filtro se aplica dentro de una sentencia *unless*, con el fin de que sólo se procese el filtro cuando hubo cambios en el feed desde la última actualización. Con esto se evita procesar entradas viejas, aumentando así la eficiencia.
El símbolo de exclamación (!) como sufijo del método, indica (por convención) que se modifica directamente el parámetro, en vez de devolver una copia modificada. La implementación del método en sí es la siguiente:

```
def apply_filter! feed
  keywords = filter.keywords.split(',')
  entries = feed.entries.select do |entry|
    if filter.list_type # whitelisting
      entry_contains_any_keyword?(entry, keywords)
    else # blacklisting
      !entry_contains_any_keyword?(entry, keywords)
    end
  end
  feed.entries = entries
end

def entry_contains_any_keyword? entry, keywords
  keywords.any? do |keyword|
    entry.title.include?(keyword) ||
    entry.summary.include?(keyword) ||
    entry._?.content._?.include?(keyword)
  end
end
```

Lo que hace este método es tomar todas las palabras clave y revisar en cada entrada por separado si el título, resumen o contenido contienen alguna de las palabras clave. En caso de tratarse de un filtro de tipo *lista blanca*, las entradas que cumplen con el criterio son seleccionadas. En cambio si se trata de un filtro de tipo *lista negra* las entradas que cumplen con el criterio son rechazadas y sólo se seleccionan las que no cumplen.

Finalmente se asigna las entradas sobrantes al arreglo de entradas del feed, para que estas puedan ser grabadas en la base de datos.
En adelante todos los feeds serán filtrados, durante su actualización, cuando su filtro esté activo.

5.4.4. Iteración 4

5.4.4.1. Gestión de notificaciones

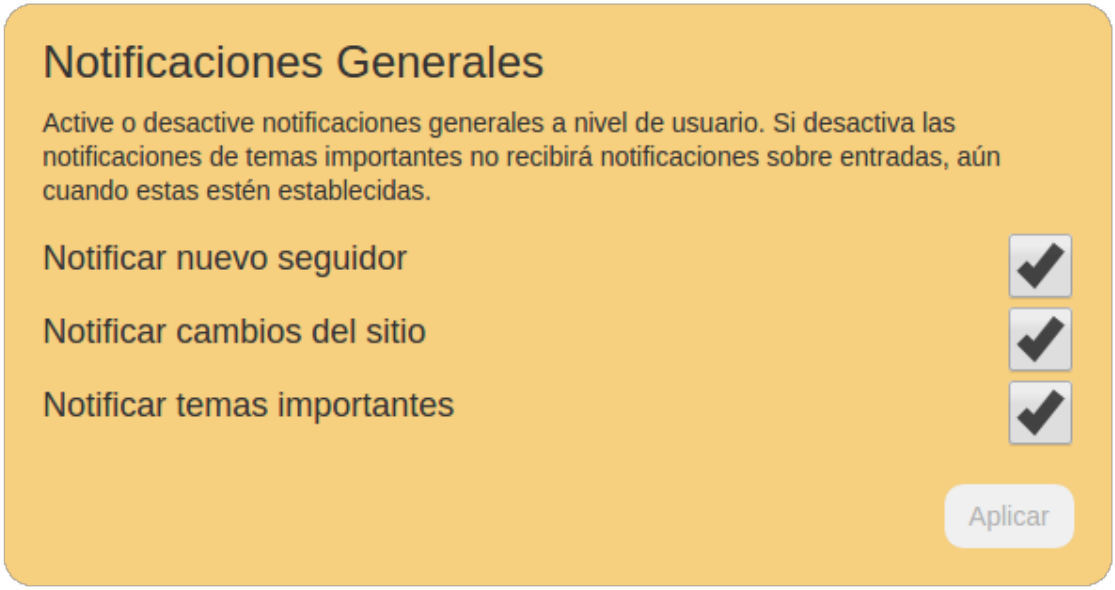
Las notificaciones se crean con el fin de alertar al usuario acerca de la aparición de una palabra clave dentro de la entrada de un feed específico. Cada notificación se crea a nivel de un feed específico. Las notificaciones llegan al usuario vía email y contienen las entradas que cumplen con el criterio de notificación. El criterio de cada notificación se evalúa con posterioridad al filtrado de cada feed, para evitar que el usuario reciba notificaciones de feeds que luego no se almacenan en la aplicación.
Además de las notificaciones a nivel de feeds, también existen notificaciones generales que pueden ser activadas o desactivadas en la sección de notificaciones.
Las notificaciones generales son:

1. Nuevo seguidor: Se notifica cada vez que otro usuario decide seguir al usuario objetivo.
2. Actualizaciones de página: Avisa al usuario si se realizaron cambios a nivel de la aplicación, como por ejemplo en las condiciones de uso o características y funcionalidades nuevas.
3. Temas importantes: Activar/Desactivar notificaciones de feeds en general.

Las notificaciones se envían con la misma frecuencia con la que se actualizan las entradas de los feeds. El mismo proceso que corre en background para actualizar las entradas se ocupa de enviar las notificaciones en caso de haber alguna. Para esto se juntan todas las notificaciones y, una vez realizadas las actualizaciones, se vuelcan todas en un mail para ser enviadas al usuario correspondiente.

5.4.4.1.1. Alta de notificaciones

El alta de notificaciones a nivel de feeds específicos sigue mayormente la misma lógica que el alta de filtros.
Primero requiere que el usuario seleccione un feed de la lista de selección. Una vez realizada la selección, se envía la información del feed al servidor, y en respuesta se despliega el formulario para crear una notificación. El formulario contiene un campo de palabras que funciona con el plugin de etiquetas mencionado para filtros en 5.4.3.1.1 (bootstrap-tagsinput-rails). El otro campo, representado por un checkbox, simplemente cumple la función de activar y desactivar la notificación, al igual como se realizó con los filtros en la iteración anterior.
La parte distintiva de la sección de notificaciones, frente a la de filtros, es el manejo de notificaciones a nivel de usuario que se ilustra a continuación:



En la captura de pantalla se puede observar que se pueden activar notificaciones a nivel general. “Notificar temas importantes” se refiere a las notificaciones a nivel de feed en general. Es decir que por más que se tengan las notificaciones a nivel de feed activadas, si esta opción está desactivada no se recibirán notificaciones a nivel de feed. Esto es útil para no tener que desactivar todas las notificaciones individualmente en caso de no querer recibir emails esporádicamente.
Además de las notificaciones de feeds, también se tiene un checkbox para ser informado sobre usuarios que siguen a la cuenta propia (“Notificar nuevo seguidor”), y otro para notificaciones acerca de actualizaciones del sitio (“Notificar cambios del sitio”). Este último tipo de notificación no se implementará, dado que los cambios en el sitio deberían ser informados manualmente por un desarrollador o administrador. En este caso sólo se guarda la información para saber a futuro, cuales son los usuarios que desean ser informados sobre los cambios del sitio.

A nivel de implementación las notificaciones a nivel de feed se encuentran asociadas directamente a los feeds mediante el modelo Notification. Las notificaciones generales están asociadas al usuario a través de atributos propios del usuario.

5.4.4.1.2. Baja de notificaciones

Para la baja de notificaciones se sigue el mismo mecanismo que se desarrolló para los filtros. Para más información ver inciso 5.4.3.1.2.

5.4.4.1.3. Modificación de notificaciones

Para modificar las notificaciones se sigue el mismo mecanismo que se desarrolló para los filtros. Para más información ver inciso 5.4.3.1.3.

5.4.4.2. Notificar

5.4.4.2.1. Notificaciones de feeds

El mecanismo de notificación se extiende a través del modelo Feed, el worker FeedWorker, el mailer NotificationMail y las vistas renderizadas por el mailer.
Ante todo, un **worker** es simplemente una clase de Ruby que se ejecuta en background (es decir, fuera del proceso principal) a través de la gema Sidekiq (ver 4.1.1.4).
Un **mailer** es otra clase nativa de Rails que sirve para enviar mails en forma automatizada. El mailer se instancia, se ejecuta un método que renderiza un mail con los parámetros necesarios, y finalmente se envía dicho mail ejecutando su método `deliver`.

Al actualizar las entradas de un feed, se requiere obtener todas aquellas que sean nuevas y cumplan con el criterio de notificación preexistente. La actualización de las entradas se lleva a cabo a través del FeedWorker (ver app/workers/feed_worker.rb) ejecutando el método `update_entries` de cada instancia de Feed. Por lo tanto, se decide obtener las notificaciones en el método mencionado.
El código correspondiente es el siguiente (ver app/models/feed.rb):

```
def update_entries
  feed = Feedjira::Feed.fetch_and_parse(url)
  raise 'Feed format issue' if feed.is_a? Numeric
  notification_entries = []
  unless self.etag == feed.etag && self.last_modified ==
feed.last_modified
    apply_filter!(feed) if filter._?.active
    user = User.find(user_id)
    if notification._?.active
      notification_entries = search_notification_entries(feed, user)
    end
    self.etag = feed.etag
    self.last_modified = feed.last_modified
    Entry.add_entries(feed.entries, self.id, user_id)
```

```
  end
  notification_entries
end
```

El código agregado está marcado en letra **verde y negrita**. Los cambios anteriores se pueden interpretar de la siguiente manera: Se crea un arreglo de entradas de notificación, se obtienen todas las entradas correspondiente a través del método (aún no implementado) `search_notification_entries(feed, user)` y finalmente se devuelve dicho arreglo. La implementación de `search_notification_entries` es la siguiente:

```
def search_notification_entries feed, user
  keywords = notification.keywords.split(',')
  notification_entries = []
  feed.entries.each do |entry|
    if !Entry.exists?(guid: entry.id) &&
entry_contains_any_keyword?(entry, keywords)
      notification_entries << entry
    end
  end
  notification_entries
end
```

El método anterior, obtiene las palabras clave del objeto de Notification asociado en un arreglo. Luego crea un arreglo para guardar todas las entradas que se deben notificar. Finalmente se averigua a través del método `contains_any_keyword?(entry, keywords)` si al menos una de las palabras clave se encuentran en el texto de la entrada.
`entry_contains_any_keyword?` revisa los atributos `title, summary y content` (en caso que esté presente) de la entrada.

El próximo paso a realizar es modificar la clase `FeedWorker` (ver app/workers/feed_worker.rb) de manera que obtenga las entradas para las notificaciones y finalmente llame a NotificationMailer (aún inexistente) para enviar las notificaciones por mail al usuario correspondiente.
Por lo tanto se agregan las siguientes líneas al método `perform` de `FeedWorker`:

```
def perform
  User.find_each(batch_size: 10) do |user|
    notifications = []
```

```
    user.feeds.each do |feed|
      begin
        entries = feed.update_entries
        unless !user.notify_important_topics || entries.empty?
          notifications << {feed: feed, entries: entries}
        end
      rescue Exception => e
        logger.debug "FeedWorker:/n#{e.message}"
      end
    end
    unless notifications.empty?
      NotificationMailer.notify_entries(user, notifications).deliver
    end
  end
end
```

El método perform es invocado automáticamente por Sidekiq con periodicidad de 15 minutos (para más información acerca de este funcionamiento, ver inciso 5.4.1.3.1).
Los cambios reflejan la asignación de las entradas (a ser notificadas) a un arreglo, luego de ejecutarse el método `update_entries` sobre el objeto de Feed. Después, en caso de estar activadas las notificaciones a nivel general y no contar con un arreglo de entradas vacío, se crea un objeto Hash por cada feed que tenga entradas a notificar. **Hash** es una clase nativa de Ruby cuyas instancias son estructuras de tipo clave-valor. Con esto se guarda temporalmente cada feed asociado con sus entradas. Finalmente, en caso de haber creado al menos un Hash, se llama el método `notify_entries` de NotificationMailer para crear el mail, y luego sobre ese mail se ejecuta el método `deliver` para enviarlo.

Por último se tiene que crear el mailer. Esto se logra mediante el comando:
`rails g mailer NotificationMailer notify_entries`
Con este comando se genera la clase NotificationMailer en app/mailers/notification_mailer.rb. Además se genera también la vista en formato de texto que renderiza el mail (app/views/notification_mailer/notify_entries.text.haml). El archivo se renombra a notify_entries.html.haml para que se puedan agregar estilos y se muestre el contenido como una página HTML. Como parte del comando ejecutado, también se genera un archivo de testing (spec/mailers/notification_mailer_spec.rb) que, como se mencionó en 5.4.1.2, no se utilizarán en este proyecto.

Con las adaptaciones pertinentes, la clase NotificationMailer queda implementada de la siguiente manera:

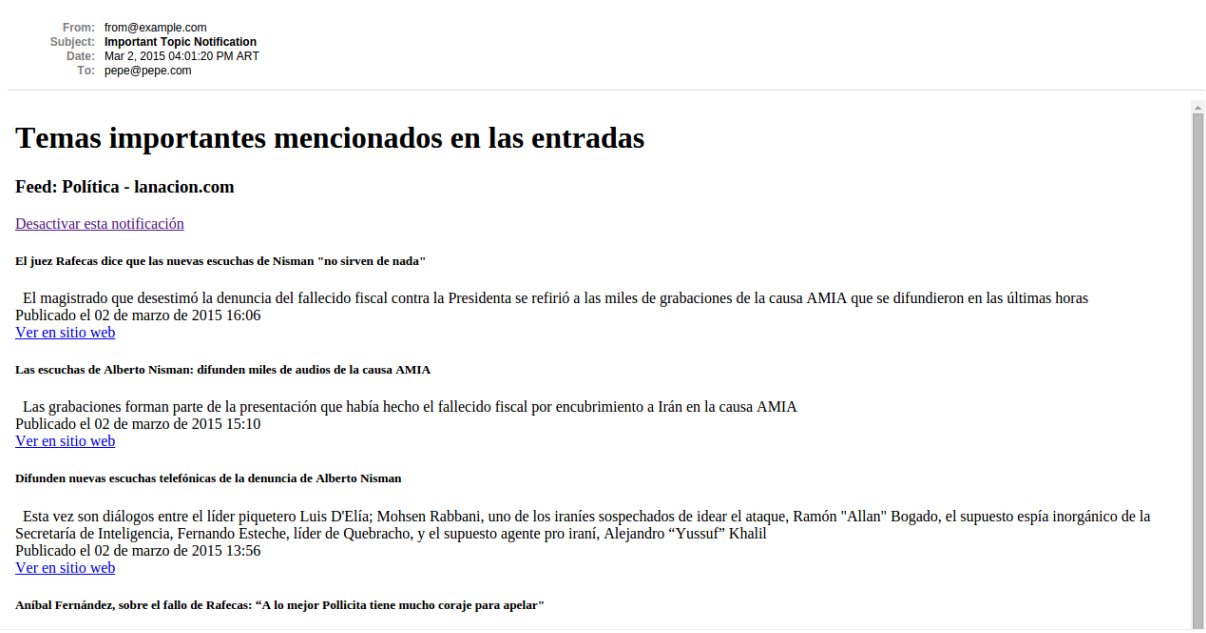
```
class NotificationMailer < ActionMailer::Base
  add_template_helper(EntriesHelper)
  default from: "from@example.com"

  def notify_entries user, notifications
    @user = user
    @notifications = notifications
    mail(to: @user.email, subject:
I18n.t('notification_mailer.notify_entries.important_topic_notification'
))
  end
end
```

Se agrega la línea `add_template_helper(EntriesHelper)` para poder utilizar los mismos métodos para formatear las entradas, que aquellos utilizados en la vista de entradas. El método `notify_entries` simplemente toma el usuario y las notificaciones como parámetros y ejecuta el método `mail` pasando la dirección de mail del usuario.

El método `mail` se ocupa de renderizar la vista homónima del método `notify_entries`, es decir notify_entries.html.haml, que se generó automáticamente. La clase I18n proviene de una gema que se utiliza para manejar las traducciones. Esto se explica con mayor detalle en la sección 4.1.1.1. Por lo tanto sólo falta crear dicha vista, para mostrar el contenido del mail correctamente. El código correspondiente se puede ver en app/views/notification_mailer/notify_entries.html.haml. No se entrará a los detalles del marcado porque no aporta a la funcionalidad, que es el principal punto de enfoque de esta Tesina. A grandes rasgos se puede mencionar que la vista consiste en un título, y luego por cada notificación muestra el título del feed con todas las entradas que cumplen el criterio de notificación del feed. Cada entrada incluye un acceso a la página correspondiente. Además cada feed cuenta con un acceso a la configuración de la notificación correspondiente a dicho feed. Esto último se incluye para el caso en que un usuario desee desactivar o modificar una notificación particular en forma rápida.

Con el fin de poder ver los mails con sus notificaciones se debería llevar a cabo una serie de configuraciones de un servidor SMTP (entre otras opciones). Para demostrar el envío y la recepción de mails en forma rápida se ha decidido utilizar una gema, llamada Letter Opener, que permite abrir automáticamente cada mail en una nueva pestaña del navegador en el momento en que aparecen. Por lo tanto esta gema permite probar el funcionamiento correcto de las notificaciones.

A continuación se muestra una captura de pantalla de ejemplo de cómo Letter Opener muestra un email de notificación. A su vez se puede apreciar la visualización final del mail, el cual es independiente de la herramienta que lo muestre.



5.4.4.2.2. Notificaciones de seguimiento

Por último, falta implementar el código necesario para alertar al usuario cuando está siendo seguido por otro usuario. La implementación consiste en agregar un método a NotificationMailer, con el fin de mandar un mail al usuario seguido. El evento al que se debe recurrir para saber cuándo notificar al usuario, es la creación de una nueva relación de seguimiento entre los usuarios. Para esto se ha implementado el modelo de relación Following (ver app/models/following.rb). Por lo tanto, se opta por crear un método y ligarlo al Callback de after_create de Following, de manera que el modelo de relación mismo se ocupe de enviar el mail cuando es creado, bajo la condición de que el usuario que está siendo seguido quiera ser notificado.

En NotificationMailer (app/mailers/notification_mailer.rb) se agrega el método siguiente:

```
def notify_following follower, followed
  @follower = follower
  @followed = followed
  mail(to: @followed.email, subject:
I18n.t('notification_mailer.notify_entries.following_notification'))
end
```

Como se mencionó anteriormente para el caso de `notify_entries`, el método `mail` renderiza una vista homónima al método en que se ejecuta. Es decir, se renderizará una vista llamada notify_following.html.haml en este caso. Dicha vista se crea en el directorio app/views/notification_mailer. En la vista se muestra simplemente un aviso al usuario seguido, de que otro usuario lo está siguiendo, además del nombre y el email de este último usuario. Además, habrá un link que podrá llevar al usuario a la sección de seguimiento de la aplicación. Para más información ver el código correspondiente.

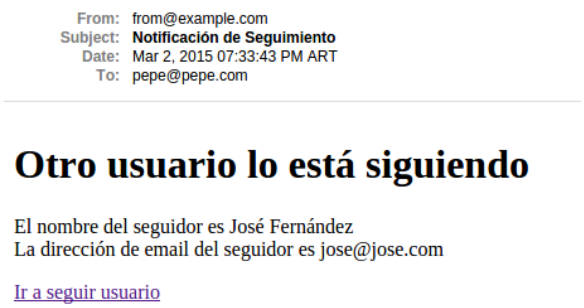
Habiendo implementado el método de notify_following de NotificationMailer, así como la vista correspondiente, sólo queda por crear el callback necesario para lograr que al crearse una relación de seguimiento, automáticamente se envíe el mail al usuario seguido. Esto se realiza en app/models/following agregando el código siguiente (marcado con **verde y negrita**):

```
class Following < ActiveRecord::Base
  acts_as_tenant :user
  after_create :notify_following, if: Proc.new {
followed.notify_new_follower }
[...]
  private
    def notify_following
      NotificationMailer.notify_following(user, followed).deliver
    end
end
```

El callback se asocia con el método `notify_following` a través del método `after_create` heredado de ActiveRecord::Base. Además se agrega la condición: `if: Proc.new { followed.notify_new_follower }` logrando que el método notify_following sólo se ejecute cuando el usuario seguido tenga la opción de notificación correspondiente activa.

Como se puede ver `notify_following` no hace mas que llamar al método necesario de NotificationMailer con los parámetros establecidos y luego llama el método `deliver` sobre el mail que devuelve.

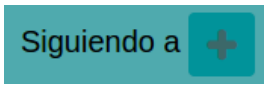
Concluimos así las notificaciones de seguimiento de usuarios. Abajo se muestra una captura de pantalla de ejemplo, para poder ver como se visualizan los mails de este tipo.



5.4.5. Iteración 5

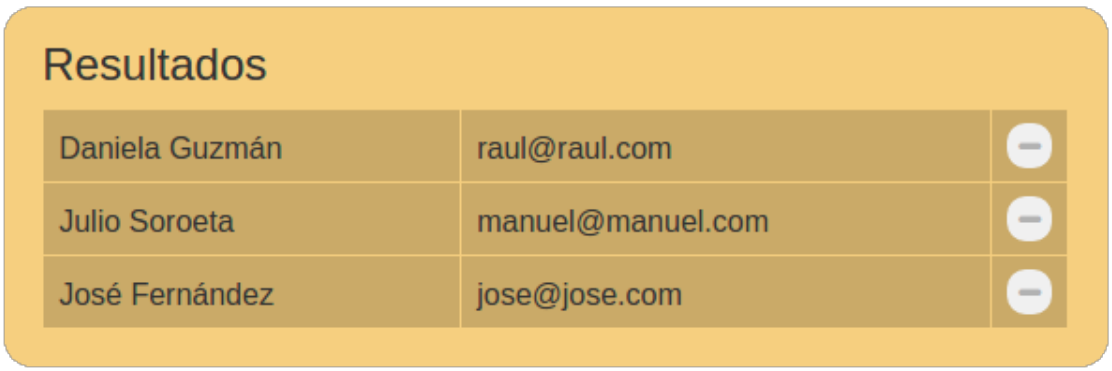
5.4.5.1. Buscar usuarios

Se accede a la vista de “seguir usuario” a través del botón ubicado inmediatamente por encima del listado de usuarios seguidos (ver imagen a continuación).



Para buscar usuarios se cuenta con un formulario construido con la gema Simple Form (ver 4.1.1.6) que se ha mencionado anteriormente. Se requiere la búsqueda mediante 2 campos del modelo de usuario (User): Nombre y email. La búsqueda misma se realiza utilizando la gema Ransack (ver 4.1.1.9) que permite simplificar considerablemente la implementación de esta funcionalidad. Esta última es la misma gema que se ha utilizado en la vista de inicio para filtrar el listado de entradas de los feeds. A continuación se muestra una captura de pantalla del formulario.

Los resultados se muestran en una tabla ubicada por debajo del formulario de búsqueda. Consiste en una simple tabla HTML de 3 columnas, conteniendo el nombre de cada usuario, la dirección de email y un botón para agregar/quitar cada usuario a/de nuestra lista de seguidos. Ver figura de abajo.



La búsqueda se implementa de la siguiente manera. En el controlador FollowingsController (ver app/controllers/followings_controller.rb) generado mediante el comando `rails g controller followings` se crea la acción (método) `index`:

```
def index
  @users, @q = FollowingsService.search_users current_user, 50, params
  respond_to do |format|
    format.html
    format.js
  end
end
```

FollowingsService es una clase que se ocupará de la lógica de búsqueda, con el fin de aliviar la cantidad de código que aparece en el controlador. Si se observa los demás controladores se podrá observar que se ha tomado este enfoque hasta el momento en todos los controladores, aún en aquellos casos en que no hubo mucho código para agregar. Esto último se llevó a cabo con el fin de asegurar que, en caso de agregar lógica en un futuro, este no ocupe espacio en el controlador. Además, agregando un servicio a cada controlador, se logra una estructura común en el manejo de la lógica en todos los controladores.
Por defecto si no se proveen parámetros al método `search_users`, se obtiene todos los usuarios, excepto el propio. Para asegurar que el propio usuario quede descartado de los resultados se implementa un scope que devuelve todos los usuarios salvo el actual. Un *scope* es simplemente un método de clase, que se utiliza en modelos para realizar consultas a la base de datos. Se puede imaginar un scope como si fuera un filtro que se ejecuta sobre la base de datos a través del modelo. La sintaxis particular del scope mencionado es `scope :except, ->(user) { where('id != ?', user) }` y se encuentra en app/models/user.rb.
Una vez obtenidos los usuarios sólo queda renderizar la vista de index ubicada en app/views/followings/index.html.haml.
La parte del controlador que se ocupa de renderizar dicha vista es `format.html`, que por defecto renderiza la vista homónima al nombre del método (en este caso, index.html.haml).
La primera vez que se carga la página se utiliza la vista en formato HTML. Para las búsquedas subsecuentes se utiliza la respuesta `format.js` con el fin de no refrescar la página completa, sino sólo la tabla de resultados. Esto se logra, como se ha mencionado en iteraciones anteriores, mediante la manipulación de la página con JavaScript (JQuery, ver 4.3.1).

5.4.5.2. Seguir usuarios

Para seguir usuarios se debe seleccionar simplemente el botón con el símbolo “+” dentro de la tabla de resultados. En caso de haber un símbolo “-”, significa que el usuario adyacente ya se encuentra en el listado de usuarios que están siendo seguidos. Vale aclarar que no sólo cambia el símbolo, sino también el vínculo subyacente (que apunta a la acción de `create` o `destroy` dependiendo del caso). El botón mencionado para agregar/quitar los usuarios va a cambiar remotamente de un símbolo al otro vía AJAX.
Dado que en la aplicación se tiene el listado de usuarios seguidos directamente integrado en el layout, se deberá actualizar dicho listado a la par que el contenido de la vista. Esto significa que en la misma respuesta de JavaScript se tendrá que contemplar código que maneje el listado del layout. Por razones de complejidad innecesaria no se entrará a los detalles del código resultante. El código correspondiente se ubica en app/views/followings/create.js.erb, y para simplificar su entendimiento se puede separar la funcionalidad en 2 bloques:

1. Se ejecuta un helper (`toggle_follow_button`) para obtener el botón de agregar/quitar con el símbolo que corresponda según si existe o no la relación de seguimiento. Luego se inserta el botón generado en el lugar donde se encuentra el botón obsoleto en la vista de “seguir usuario”.
2. En el listado de usuarios seguidos (layout), se limpia el listado en caso que este no tuviera usuarios. Luego se renderiza un partial para obtener el elemento HTML que alberga el usuario. Finalmente se agrega el usuario al listado.

5.4.5.3. Dejar de seguir usuarios

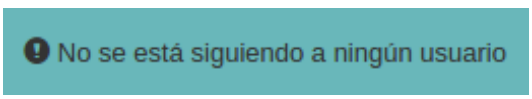
Existen 2 lugares en los que se puede dejar de seguir a los usuarios:

1. En la vista de “seguir usuario”, a través del botón con el símbolo “-” en la tabla de resultados.
2. En el listado de usuarios seguidos del layout, a través del botón junto al nombre del usuario.

En ambos casos se ejecuta la misma acción (`destroy`) en el controlador FollowingsController, y la respuesta es la misma. Por lo tanto, así como se ha hecho en el caso de `create`, para esta respuesta también se tendrá que tener en cuenta el usuario que se quiere eliminar del layout (que siempre estará presente), así como aquel de la vista de “seguir usuarios” (que posiblemente no figure en pantalla, al poder eliminar usuarios desde el listado de layout estando en cualquier vista). La siguiente imagen permite observar esta dificultad con mayor claridad.

Por lo anterior, puede parecer que este problema requerirá de una solución sustancialmente más complicada que en el caso de `create`. Sin embargo, JQuery (ver 4.3.1) facilita el problema debido a que si no se encuentra el lugar en la página, en el que se quiere modificar un elemento, la modificación no se lleva a cabo. Es así como se puede escribir código sin sentencias de flujo de control para limitar la ejecución de instrucciones frente a ciertas condiciones. Es decir, no es necesario evaluar si se está en la vista de “seguir usuarios”, para saber si se debe ejecutar el código que cambia el estado del botón de agregar/quitar de un usuario particular.
La respuesta de JavaScript de la acción `destroy` en el controlador FollowingsController consiste de 2 bloques:

1. Ídem primera parte de create.js.erb. (ver primer bloque en 5.4.5.2)
2. En el listado de usuarios seguidos (layout), se elimina el usuario particular, y en caso de no quedar más usuarios en dicho listado, se despliega el siguiente mensaje:



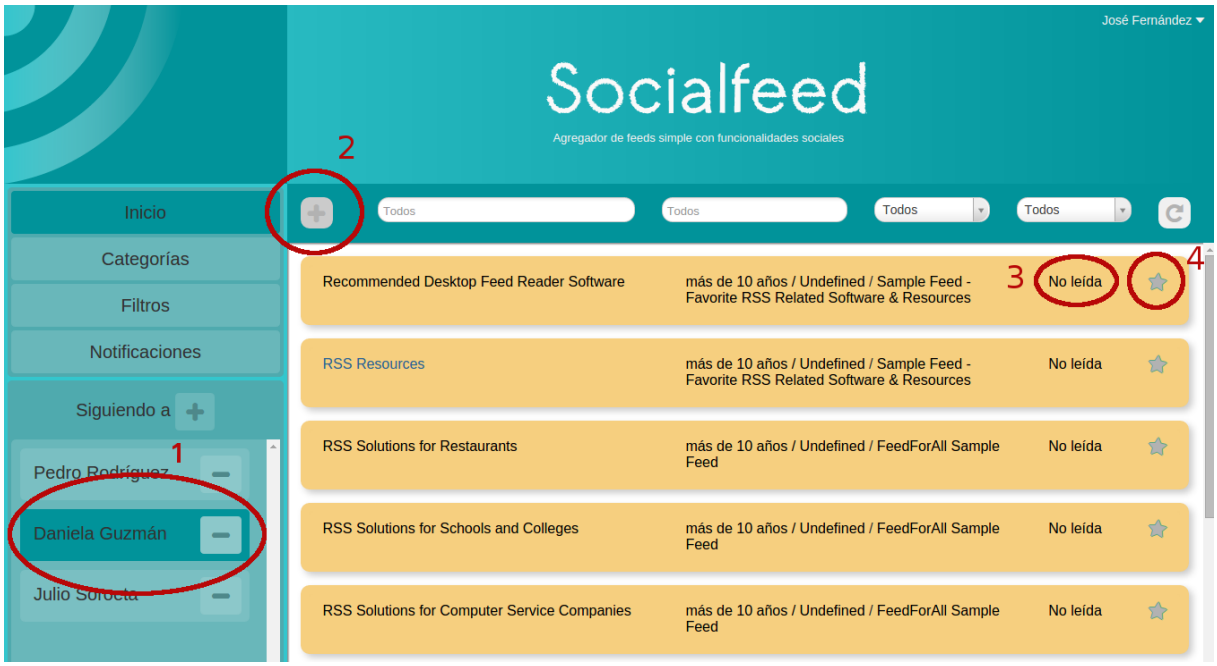
5.4.5.4. Ver feeds de usuarios seguidos

Una vez que se agregó un usuario al listado de usuarios seguidos (en el layout), se puede seleccionar el nombre del usuario para ver las entradas del mismo. De esta manera, el nombre del usuario representa un enlace que está ligado a la acción `index` del controlador EntriesController.
El problema es el siguiente: Con el fin de separar los datos de los usuarios sin tener que crear una base de datos o tablas separadas para cada uno, se ha utilizado la gema Acts As Tenant (ver

4.1.1.8) que separa todos los datos mediante una asociación estricta al usuario. Teniendo esto en cuenta, no se puede acceder a los datos de otro usuario, si no es desactivando la restricción impuesta por la gema anterior. Afortunadamente la gema permite cambiar el usuario manualmente dentro de la acción de un controlador. Esto facilita el acceso a las entradas de cualquier usuario y se puede implementar agregando el siguiente código (en **verde y negrita**) a la acción `index` de EntriesController:

```
def index
  if params[:user_id]
     @user = User.find(params[:user_id])
     ActsAsTenant.current_tenant = @user
  end
  [...]
end
```

De esta forma se puede enviar el id del usuario cuyas entradas se quieren observar, y se obtiene el resultado esperado. El vínculo correspondiente estará definido por `link_to entries_path(user_id: followed_id)`. Notar que se está pasando un símbolo (:user_id) como parámetro para indicar el usuario del cual se desea ver las entradas.
Queda un inconveniente por resolver: Si se renderiza la vista sin cambios algunos, el usuario actual podrá modificar el estado de las entradas del usuario seguido (marcar como leído/no leído o favorito/ no favorito). Para evitar esto se debe deshabilitar los botones y vínculos correspondientes cada vez que se está accediendo a las entradas de otro usuario. En este caso se optó por deshabilitar los botones y vínculos por medio de CSS cuando esté asignado un usuario a una variable particular. Sin embargo, esto aún permitiría enviar una petición HTTP en forma manual al servidor (incluso modificando atributos de los botones/vínculos), por lo cual esta solución no sería la más adecuada. De todas formas, por más que se intentara modificar una entrada sobrepasando las restricciones de CSS, no se podría modificarla debido a que Acts As Tenant no permitiría el acceso. Simplemente no se encontraría la entrada con el id correspondiente y se produciría un error interno, sin significar esto la caída de la aplicación.
Abajo se muestra una captura de pantalla como ejemplo de cuando se tiene seleccionado un usuario seguido para visualizar sus entradas. En la imagen se marcan las partes que cambian entre la vista de entradas del propio usuario y la de los usuarios seguidos. Se podrá ver que el usuario seleccionado queda resaltado en el listado de seguidos, y que si bien los botones de cambio de estado (leído/no leído y favoritos) y agregar feeds están desactivados, aún se permite filtrar las entradas por lo criterios originales.



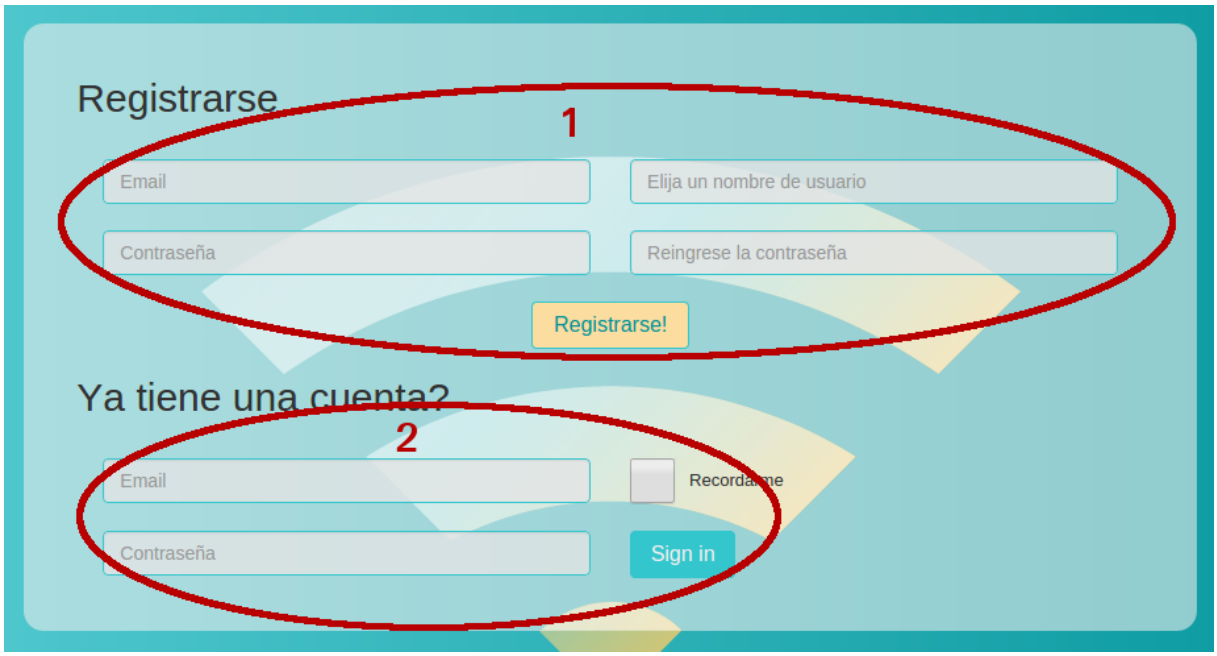
Referencias:

1. Usuario seleccionado en listado de usuarios seguidos. Sirve de indicio para saber el usuario al que corresponden las entradas. En caso de no haber usuario seleccionado, las entradas visualizadas son las propias.
2. “Agregar feed” desactivado, para que no se intente agregar feeds a otros usuarios.
3. Vínculo leído/no leído desactivado. Visualmente no habrá diferencias, sin embargo seleccionar el vínculo no tendrá efecto alguno.
4. Botón de favoritos deshabilitado. Las estrellas de favoritos aparecen en un tono gris que simboliza la inactividad de los botones.

5.4.6. Iteración 6

5.4.6.1. Formularios de Landing

En esta última iteración se creará la página inicial de la aplicación conocida como “landing”. Esta es la primer página visible de la aplicación, antes de crear un usuario o acceder a la sesión de un usuario preexistente. El landing se mostrará únicamente cuando el usuario no se encuentre conectado a la hora de acceder a la aplicación. Funcionalmente no implica mayores desafíos y consiste únicamente en 2 formularios generados automáticamente por la gema Devise (ver 4.1.1.2). A continuación se muestran ambos formularios con una breve explicación de cada uno:



1. Registrarse (ver app/views/devise/registrations/_new.html.haml): Formado por los campos: Email, Nombre, Contraseña y Confirmación de contraseña. Al enviar este formulario (por medio del botón “Registrarse!”) se crea el usuario y se entra a la sesión correspondiente.
2. Entrar (ver app/views/devise/sessions/_new.html.haml): Consta de los campos Email y Contraseña, además de un checkbox “Recordarme”. Los primeros 2 campos son necesarios para crear una sesión de usuario. El checkbox es opcional, en caso de estar seleccionado la sesión sigue abierta aún cuando se cierra la página. Esto hace posible ingresar directamente al menú principal de la aplicación, una vez que se ha iniciado sesión anteriormente.

En el caso de ambos formularios los controladores, que manejan la creación de usuarios y sesiones, forman parte de la gema Devise (ver 4.1.1.2) y no necesitan ser modificados para los requisitos de este proyecto. De hecho, estos controladores se dejan modificar sólo por medio de herencia.

Al tener 2 formularios que comparten una instancia de usuario en una misma página, se presenta el siguiente problema: Antes de enviar no se presenta problema alguno, los datos del formulario son enviados a controladores diferentes, según el formulario que se envía. El inconveniente se produce cuando se debe volver a renderizar la página debido a un error producido en alguno de los campos. En este caso, ambos formularios se cargarían con los mismos datos y, por el funcionamiento de Simple Form (ver 4.1.1.6), se mostrarían los mismos errores en ambos formularios. Por ejemplo: Si se ingresa un formato de email equivocado, al enviar el formulario se descubre el error y luego se carga dicho error en ambos formularios. Para evitar que esto suceda se implementó la siguiente solución en app/views/layouts/landing.html.haml:

```
.row
  .col-xs-12
    = render 'devise/registrations/new', resource: controller_name ==
'registrations' ? resource : User.new
.row
  .col-xs-12
    = render 'devise/sessions/new', resource: controller_name ==
'sessions' ? resource : User.new
```

En verde y negrita se muestra el código adicionado para solucionar el problema de renderizar errores en ambos formularios a la vez. `controller_name` es una variable de entorno de Rails que contiene el nombre del controlador que está procesando la petición en el momento. Se está utilizando además un operador ternario cuya sintaxis es “<condición> ? <verdadero> : <falso>”. Por lo tanto, en el primer caso la operación se traduce como: Si el controlador actual es “registrations” entonces se asigna el usuario proveniente del controlador (“resource”) al formulario subyacente. En caso contrario se asigna una instancia nueva del modelo User al formulario (la cual no contiene mensajes de error alguno, dado que es una instancia nueva con variables vacías). De esta manera, la condición funciona como un semáforo entre ambos formularios.

5.4.6.2. Notificaciones Flash

En todas las vistas pueden aparecer notificaciones flash, que pueden ser de tipo notice o alert. En Rails estas se integran por defecto en la parte superior de los layouts y en principio no cuentan con estilos preestablecidos. Para volver estas notificaciones más atractivas se agregan Modals, un componente de Bootstrap utilizado previamente en el inciso 5.4.1.3.5. Esto no sólo se aplica en el layout de landing, sino que también se agrega al layout de aplicación general (en app/views/layouts/application.html.haml) para las notificaciones correspondientes.
Mediante los siguientes pasos implementamos las notificaciones con Modals:

1) Crear un partial que contenga el modal y el mensaje a mostrar en el mismo (ver app/views/layouts/_notification.html.haml).

2) Agregar el partial a los layouts (landing.html.haml y application.html.haml en app/views/layouts):

```
- if notice
  = render 'layouts/notification', message: notice, type: 'info'
- if alert
  = render 'layouts/notification', message: alert, type: 'danger'
```

3) El modal ya está creado pero requiere ser instanciado mediante JavaScript. Por eso se agrega la siguiente línea (en **verde y negrita**) a app/assets/javascripts/main.js (este archivo se ejecuta automáticamente al cargar cada página):

```
$(document).ready(function(){
  var app = new App();
  app.apply_select2();
  app.apply_tooltip();
  app.initiate_forms();
  $('#notification').modal();
});
```

Habiendo llevado a cabo esto último, ya se puede visualizar las notificaciones flash de la manera deseada. Sólo faltan ser agregados los estilos que pueden observarse en main.css.scss pero no profundizaremos en explicar este aspecto.

5.4.6.3. Responsividad

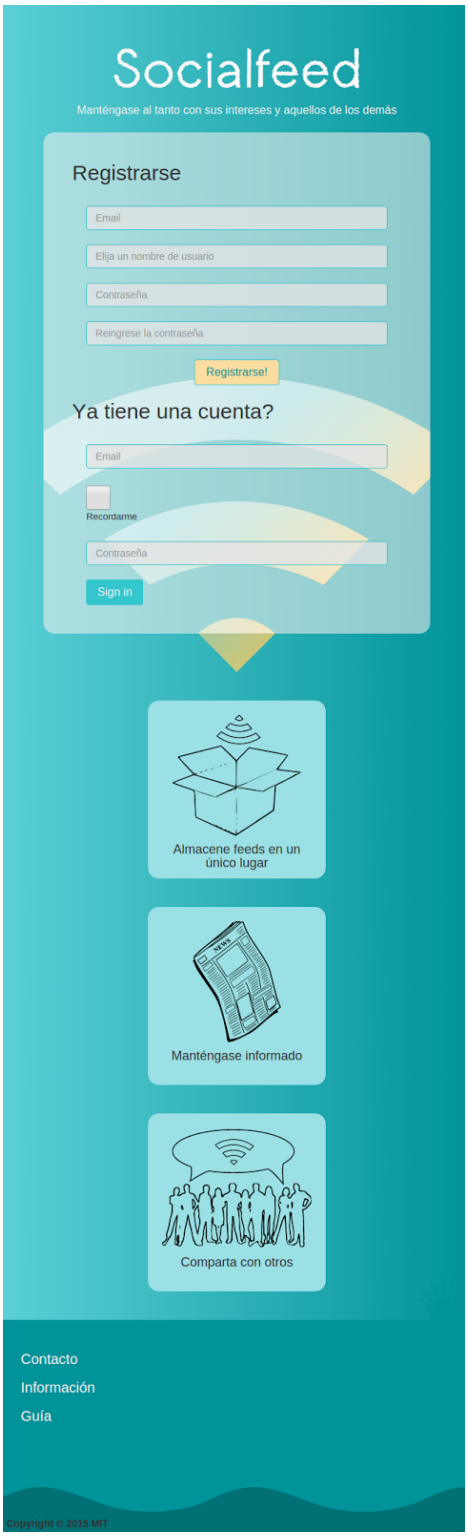
Bootstrap permite el diseño de páginas web que adaptan su contenido según el tamaño de la pantalla. El concepto que engloba esta capacidad se conoce como RWD (Responsive Web Design). No se entrará en los detalles de su implementación, pero si se aclara que se utilizaron algunos de los componentes que provee Bootstrap para lograr un diseño responsivo de la página landing. Esto puede ser probado disminuyendo o agrandando el tamaño de la ventana en la que se visualiza la página. A continuación se muestran imágenes que demuestran el comportamiento de la página según el tamaño de la pantalla:

≥ 1200 pixeles:

≥ 992 pixeles y < 1200 pixeles:

≥ 768 pixeles y < 992 pixeles:

≥ 480 pixeles y < 768 pixeles:


Socialfeed
Manténgase al tanto con sus intereses y aquellos de los demás
Registrarse
Email
Elija un nombre de usuario
Contraseña
Reingrese la contraseña
Registrarse!
Ya tiene una cuenta?
Email
Recordarme
Contraseña
Sign in
Almacene feeds en un único lugar
Manténgase informado
Comparta con otros
Contacto
Información
Guía
Copyright © 2015 MIT

< 480 pixeles:


Socialfeed
Manténgase al tanto con sus intereses y aquellos de los demás
Registrarse
Email
Elija un nombre de usuario
Contraseña
Reingrese la contraseña
Registrarse!
Ya tiene una cuenta?
Email
Recordarme
Contraseña
Sign in
Almacene feeds en un único lugar
Manténgase informado
Comparta con otros
Contacto
Información
Guía
Copyright © 2015 MIT

6. Conclusiones

La aplicación desarrollada satisface los requerimientos básicos establecidos mediante user stories. Además se cubren algunos requerimientos importantes, surgidos durante el diseño, el desarrollo e incluso luego de implementar la propuesta inicial de la aplicación. Es así como, a lo largo de esta Tesina, se reafirmó la necesidad de un proceso iterativo, que permitiera pulir y modificar características, que no se podían definir con claridad desde un comienzo.

Una ventaja de mantener un conjunto pequeño de requerimientos, es la flexibilidad que conlleva para realizar cambios durante el desarrollo. Los cambios no requieren modificar un conjunto significativo de elementos, al haberse tomado muy pocas decisiones de antemano. De esto se trata uno de los principios de Lean Software Development (ver 3.3.3). Dicha metodología sostiene que se pueden obtener beneficios cuando se aplazan los compromisos, como lo son, por ejemplo, las decisiones sobre ciertos requerimientos.

Mantener una base de requerimientos acotada, también constituye una manera de eliminar residuos o, mejor dicho, evitar su aparición y acumulación. Documentación sobre requerimientos que finalmente no se implementan o cuyo destino es dudoso, puede significar un desperdicio de tiempo y recursos.

Por otra parte, teniendo un conjunto acotado de requerimientos, se puede organizar fácilmente el desarrollo en iteraciones. Al final de cada una de estas, se puede realizar una entrega del producto parcialmente terminado, con el fin de evaluar su funcionamiento, obtener retroalimentación y ajustar el rumbo del desarrollo. En contraposición, se encuentran las metodologías en las que no se cuenta con información acerca de la satisfacción de las necesidades del cliente o usuario, hasta que no se haya finalizado el desarrollo. Cuando sucede esto, no hay forma de asegurar que se esté siguiendo el camino correcto, de manera que se depende fuertemente de un relevamiento minucioso antes de comenzar con el desarrollo de un proyecto. Aún si los requerimientos se toman con cuidado extremo, es muy probable que el cliente/usuario será enfrentado finalmente con un producto, al menos sutilmente, diferente al que se imaginó al principio.

La aplicación creada en este trabajo constituye un MVP (Minimum Viable Product) en un aspecto de mero desarrollo. Esto significa que:

- cumple con las necesidades básicas para poder funcionar.
- se desarrolló con una inversión mínima de recursos, indispensable para ser concretado.
- podría ser desplegado y perfeccionado en base a la demanda de los usuarios.

Para esto último, se debe aclarar que la mejor manera de llevar a cabo la retroalimentación, sería

integrando un mecanismo en la aplicación para estimular que los usuarios den su opinión acerca de la aplicación. Actualmente existen muchos sitios que cuentan, por ejemplo, con una etiqueta de “feedback” que se encuentra visible en todo momento, con el propósito de motivar al usuario a proveer información de ayuda para continuar con el desarrollo.

El agregador RSS/Atom de la presente Tesina se implementó como una aplicación web por 2 principales ventajas frente a aplicaciones de tipo desktop:

- No es necesario lanzar un producto terminado. La aplicación puede completarse sin que esto requiera que el usuario descargue algún contenido. Los cambios se reflejan de directamente en el sitio web, asegurando su adopción global e inmediata.
- La aplicación puede evolucionar a medida que se utiliza. Los patrones de uso de la aplicación y los comentarios de los usuarios pueden ser las principales pautas para identificar las necesidades más urgentes y así guiar el desarrollo.

Dicho esto último, es importante manejar la evolución de una aplicación con prudencia y premeditar cada cambio antes de ser realizado. La facilidad y velocidad con la que se permite introducir cambios, pueden tentar a intentar cumplir con gran parte de la demanda de los usuarios en forma indiscriminada. Sin embargo, se debe tener en cuenta que los cambios pueden ser costosos, no sólo en su implementación, sino también en el mantenimiento. Cada vez que se agrega una característica a una aplicación, esta última se vuelve más compleja y difícil de mantener. Una vez lanzada una característica no será fácil volver a quitarla, sin tener que contar con la protesta de ciertos usuarios. Además puede que los cambios requeridos por un grupo de usuarios entren en conflicto con las expectativas de otros usuarios. De esta manera, se pone en evidencia el peligro que se esconde detrás de la flexibilidad de las aplicaciones web para introducir cambios.

7. Bibliografía

7.1. Páginas

- Siepen, D. (26 de Febrero de 2014). Top 15 sites built with Ruby on Rails. Consultado el 6 de Marzo de 2015, Obtenido de https://thecoderfactory.com/posts/top-15-sites-built-with-ruby-on-rails
- Trello. (n.d.). Consultado el 6 de Marzo de 2015, Obtenido de https://trello.com/home
- It keeps getting better. (n.d.). Consultado el 8 de Marzo de 2015, Obtenido de https://enterprise.github.com/
- Git. (n.d.). Consultado el 8 de Marzo de 2015, Obtenido de http://git-scm.com/
- What Is RSS? RSS Explained. (n.d.). Consultado el 9 de Marzo de 2015, Obtenido de http://www.whatisrss.com/
- RSS vs Atom . which one is better ? (5 de Agosto de 2012). Consultado el 9 de Marzo de 2015, Obtenido de https://shafiq2410.wordpress.com/2012/08/05/rss-vs-atom-which-one-is-better/
- Agile Methodology. (n.d.). Consultado el 9 de Marzo de 2015, Obtenido de http://agilemethodology.org/
- What is Kanban? (n.d.). Consultado el 9 de Marzo de 2015, Obtenido de http://kanbanblog.com/explained/
- Waters, K. (16 de Agosto de 2010). 7 Key Principles of Lean Software Development. Consultado el 24 de Marzo de 2015, Obtenido de http://www.allaboutagile.com/7-key-principles-of-lean-software-development-2/
- Janssen, C. (n.d.). What is a Minimum Viable Product (MVP)? - Definition. Consultado el 24 de Marzo de 2015, Obtenido de http://www.techopedia.com/definition/27809/minimum-viable-product-mvp
- Ruby. (n.d.). Consultado el 9 de Marzo de 2015, Obtenido de https://www.ruby-lang.org/en/
- Ruby. (n.d.). Consultado el 9 de Marzo de 2015, Obtenido de https://www.ruby-lang.org/en/about/
- Ruby. (n.d.). Consultado el 9 de Marzo de 2015, Obtenido de https://www.ruby-lang.org/en/libraries/
- Fuchs, S., Harvey, J., Soller, S., Moore, S., & Aimonetti, M. (2008). Svenfuchs/i18n. Consultado el 9 de Marzo de 2015, Obtenido de https://github.com/svenfuchs/i18n
- Fuchs, S., & Minařík, K. (n.d.). Guides.rubyonrails.org. Consultado el 10 de Marzo de 2015, Obtenido de http://guides.rubyonrails.org/i18n.html
- Plataformatec/devise. (2009). Consultado el 9 de Marzo de 2015, Obtenido de https://github.com/plataformatec/devise

- Allured, J., Dix, P., Kirch, J., & Templin, E. (n.d.). Feedjira/feedjira. Consultado el 9 de Marzo de 2015, Obtenido de https://github.com/feedjira/feedjira
- Feedjira. (n.d.). Consultado el 9 de Marzo de 2015, Obtenido de http://feedjira.com/
- Simple, efficient message processing for Ruby. (n.d.). Consultado el 9 de Marzo de 2015, Obtenido de http://sidekiq.org/
- Perham, M. (n.d.). Mperham/sidekiq. Consultado el 9 de Marzo de 2015, Obtenido de https://github.com/mperham/sidekiq
- Svensson, T. (n.d.). Tobiassvn/sidetiq. Consultado el 9 de Marzo de 2015, Obtenido de https://github.com/tobiassvn/sidetiq
- HoboFields. (n.d.). Consultado el 9 de Marzo de 2015, Obtenido de http://www.hobocentral.net/manual/hobo_fields
- Valim, J., Da Silva, C., França, R., & Ermolovich, V. (n.d.). Plataformatec/simple_form. Consultado el 9 de Marzo de 2015, Obtenido de https://github.com/plataformatec/simple_form
- Casimir, J., Klabnik, S., Ermolovich, V., & Haines, A. (n.d.). Drapergem/draper. Consultado el 9 de Marzo de 2015, Obtenido de https://github.com/drapergem/draper
- Matthijssen, E. (2011). ErwinM/acts_as_tenant. Consultado el 9 de Marzo de 2015, Obtenido de https://github.com/ErwinM/acts_as_tenant
- Miller, E. (1 de Enero de 2011). Activerecord-hackery/ransack. Consultado el 9 de Marzo de 2015, Obtenido de https://github.com/activerecord-hackery/ransack
- Seguin, W. (n.d.). ∞The Basics of RVM. Consultado el 12 de Marzo de 2015, Obtenido de https://rvm.io/rvm/basics
- Heinemeier Hansson, D. (n.d.). Guides.rubyonrails.org. Consultado el 13 de Marzo de 2015, Obtenido de http://guides.rubyonrails.org/getting_started.html
- ASP.NET MVC Tutorial. (n.d.). Consultado el 13 de Marzo de 2015, Obtenido de http://www.w3schools.com/aspnet/mvc_intro.asp
- Heinemeier Hansson, D. (n.d.). Guides.rubyonrails.org. Consultado el 13 de Marzo de 2015, Obtenido de http://guides.rubyonrails.org/active_record_basics.html
- The basics of html. (n.d.). Consultado el 13 de Marzo de 2015, Obtenido de https://docs.webplatform.org/wiki/guides/the_basics_of_html
- Catlin, H., & Weizenbaum, N. (n.d.). About. Consultado el 13 de Marzo de 2015, Obtenido de http://haml.info/about.html
- About JavaScript. (n.d.). Consultado el 13 de Marzo de 2015, Obtenido de https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript
- JavaScript Libraries. (n.d.). Consultado el 13 de Marzo de 2015, Obtenido de http://www.w3schools.com/js/js_libraries.asp

- JQuery. (n.d.). Consultado el 13 de Marzo de 2015, Obtenido de http://jquery.com/
- Eguiluz, J. (n.d.). Capítulo 1. Introducción a AJAX (Introducción a AJAX). Consultado el 15 de Marzo de 2015, Obtenido de http://librosweb.es/libro/ajax/capitulo_1.html
- Introducing JSON. (n.d.). Consultado el 15 de Marzo de 2015, Obtenido de http://json.org/
- 1.1 Getting Started - About Version Control. (n.d.). Consultado el 15 de Marzo de 2015, Obtenido de http://git-scm.com/book/en/v2/Getting-Started-About-Version-Control
- Git. (n.d.). Consultado el 15 de Marzo de 2015, Obtenido de http://git-scm.com/
- About. (n.d.). Consultado el 15 de Marzo de 2015, Obtenido de http://git-scm.com/about
- Build software better, together. (n.d.). Consultado el 16 de Marzo de 2015, Obtenido de https://github.com/features
- About. (n.d.). Consultado el 16 de Marzo de 2015, Obtenido de http://git-scm.com/about/small-and-fast
- CSS Syntax. (n.d.). Consultado el 16 de Marzo de 2015, Obtenido de http://www.w3schools.com/css/css_syntax.asp
- Sass (Syntactically Awesome StyleSheets). (n.d.). Consultado el 16 de Marzo de 2015, Obtenido de http://sass-lang.com/documentation/file.SASS_REFERENCE.html
- Add Vector Icons to Your Website. (n.d.). Consultado el 17 de Marzo de 2015, Obtenido de http://fontastic.me/
- What is Responsive Design? (n.d.). Consultado el 17 de Marzo de 2015, Obtenido de http://www.ugurus.com/responsive-design-definition
- Bootstrap · The world's most popular mobile-first and responsive front-end framework. (n.d.). Consultado el 17 de Marzo de 2015, Obtenido de http://getbootstrap.com/
- About. (n.d.). Consultado el 17 de Marzo de 2015, Obtenido de http://getbootstrap.com/about/
- Writing user stories. (n.d.). Consultado el 17 de Marzo de 2015, Obtenido de https://www.gov.uk/service-manual/agile/writing-user-stories.html
- ETag and Last-Modified Headers¶. (n.d.). Consultado el 18 de Marzo de 2015, Obtenido de https://pythonhosted.org/feedparser/http-etag.html
- Usage. (29 de Agosto de 2011). Consultado el 18 de Marzo de 2015, Obtenido de http://www.colorschemedesigner.com/blog/usage/
- What is iterative development? - Definition Obtenido de WhatIs.com. (n.d.). Consultado el 17 de Marzo de 2015, Obtenido de http://searchsoftwarequality.techtarget.com/definition/iterative-development
- Using HTTP Methods for RESTful Services. (n.d.). Consultado el 20 de Marzo de 2015, Obtenido de http://www.restapitutorial.com/lessons/httpmethods.html

8. Anexo - Código Fuente

8.1. app/

8.1.1. assets/

8.1.1.1. javascripts/

8.1.1.1.1. categories/

8.1.1.1.1.1. categories.js

```
//= require categories/category
```

8.1.1.1.1.2. category.js

```
(function() {

  function Category() {
    console.debug(this.constructor.name + ': instance CREATED');
    this.configure();
  };

  Category.prototype.configure = function() {
    this.ui = {};
    this.ui.form_container = '.js-category-form-container';
    this.ui.list_container = '.js-categories-list-container';
  }

  Category.prototype.refresh_form = function($form) {
    var category = this;
    $(category.ui.form_container).fadeOut("slow", function() {
      $(category.ui.form_container).html($form);
      (new App()).apply_tagsinput();
      $(category.ui.form_container).fadeIn('slow');
    });
  };

  Category.prototype.refresh_list = function($list) {
    var category = this;
    $(category.ui.list_container).fadeOut("slow", function() {
      $(category.ui.list_container).html($list);
      $(category.ui.list_container).fadeIn('slow');
      $('[data-toggle="tooltip"]').tooltip({html: true});
    });
  };
```

```
  window.Category = Category;
})();
```

8.1.1.1.2. entries/

8.1.1.1.2.1. entries.js

```
//= require entries/entry

$(document).ready(function(){
        var     entry = new Entry();
        entry.initiate_category_select();
});
```

8.1.1.1.2.2. entry.js

```
(function() {

  function Entry() {
    console.debug(this.constructor.name + ': instance CREATED');
    this.configure();
  };

  Entry.prototype.configure = function() {
    this.ui = {};
    this.ui.entries = '.js-entries';
    this.ui.entry = '.js-entry';
    this.ui.read_link = '.js-read-link';
    this.ui.favorite_link = '.js-favorite-link';
    this.ui.add_button = '#add-feed-modal';
    this.ui.category_form = '.js-category-form';
    this.ui.search_form = '.js-entries-search-form';
  }

  // Manage Category select
  Entry.prototype.initiate_category_select = function() {
    $(this.ui.category_form).on('change', 'select', function(){
      $(this).closest('form').submit();
    });
  };

  // refresh entries search form
  Entry.prototype.refresh_search_form = function($form) {
    var entry = this;
    $(entry.ui.search_form).fadeOut("slow", function() {
      $(entry.ui.search_form).html($form);
      $(entry.ui.search_form).fadeIn('slow');
      $('select').select2();
```

```
    });
  };

  // refresh entries
  Entry.prototype.refresh_entries = function($entries) {
    var entry = this;
    $(entry.ui.add_button).modal('hide');
    $(entry.ui.entries).fadeOut("slow", function() {
      $(entry.ui.entries).html($entries);
      $(entry.ui.entries).fadeIn('slow');
    });
  };

  window.Entry = Entry;
})();
```

8.1.1.1.3. feeds/

8.1.1.1.3.1. feed.js

```
(function() {

  function Feed() {
    console.debug(this.constructor.name + ': instance CREATED');
    this.configure();
  };

  Feed.prototype.configure = function() {
    this.ui = {};
    this.ui.add_button = '#add-feed-modal';
  }

  Feed.prototype.refresh_form = function($form) {
    // Manage entry submit button
    $(this.ui.add_button).find('.modal-content').html($form);
    $(this.ui.add_button).modal('show');
  };

  window.Feed = Feed;
})();
```

8.1.1.1.4. filters/

8.1.1.1.4.1. filters.js

```
$(document).ready(function(){
  var app = new App();
  app.initiate_selection_form();
});
```

8.1.1.1.5. followings/

8.1.1.1.5.1. followings.js

```
$(document).ready(function(){
  var app = new App();
  app.initiate_selection_form();
});
```

8.1.1.1.6. application.js

```
// This is a manifest file that'll be compiled into application.js, which will include all the files
// listed below.
//
// Any JavaScript/Coffee file within this directory, lib/assets/javascripts, vendor/assets/javascripts,
// or vendor/assets/javascripts of plugins, if any, can be referenced here using a relative path.
//
// It's not advisable to add code directly here, but if you do, it'll appear at the bottom of the
// the compiled file.
//
// WARNING: THE FIRST BLANK LINE MARKS THE END OF WHAT'S TO BE PROCESSED,
ANY BLANK LINE SHOULD
// GO AFTER THE REQUIRES BELOW.
//
//= require jquery
//= require jquery_ujs
//= require twitter/bootstrap
//= require bootstrap-tagsinput
//= require select2
//= require main
//= require_tree .
```

8.1.1.1.7. bootstrap.js.coffee

```
jQuery ->
  $("a[rel~=popover], .has-popover").popover()
  $("a[rel~=tooltip], .has-tooltip").tooltip()
```

8.1.1.1.8. main.js

```
(function(){
  function App() {
    App.prototype.apply_select2 = function() {
      $('select').select2();
    };
    App.prototype.apply_tooltip = function() {
      $('[data-toggle="tooltip"]').tooltip({html: true});
    };
    App.prototype.apply_tagsinput = function(){
```

```
    $('[data-role="tagsinput"]').tagsinput();
  };
  App.prototype.initiate_forms = function() {
    // Manage entry submit button
    $('.js-form-container').on('click', '.js-submit', function(){
      $(this).closest('form').submit();
    });
    $('.js-form-container').on('keydown', '.form-control', function(e){
      if(e.which == 13){
        e.stopImmediatePropagation();
        $(this).closest('form').submit();
      }
    });
  };
  App.prototype.initiate_selection_form = function() {
    $('.js-selection-form').on('change', 'select', function(){
      $(this).closest('form').submit();
    });
  };
  App.prototype.refresh_content = function($container_selector, $content, callback) { // TODO:
Apply this to all views and remove repeated code
    $container_selector.fadeOut("slow", function() {
      $container_selector.html($content);
      $container_selector.fadeIn('slow');
      if(typeof callback !== 'undefined'){
        callback();
      }
    });
  };
 }
 window.App = App;
})();

$(document).ready(function(){
 var app = new App();
 app.apply_select2();
 app.apply_tooltip();
 app.initiate_forms();
 $('#notification').modal();
});
```

8.1.1.2. stylesheets/

8.1.1.2.1. _sass_defaults.scss

```
/*
================================================================================
  Sass
```

```
================================================================================
*/

/* ================================
  Font Styles
  ================================ */
$font: verdana, arial, sans-serif;
$fsbody: 16px;
$fs-h1: 48px;
$fs-h2: 36px;
$fs-h3: 28px;
$fs-h4: 20px;
// $w: 940px;

/* ================================
  Colors
  ================================ */
// $bg: #f1f2f7;
// $dbg: #292d30;

$white: #f0f0f0;
$black: #333333;
$gray: #b2b2b2;
$icongray: #4e888a;
$dgray: #b2b2b2;

// icons
$yellow: #ffba2b;
$lyellow: #fbdda0;
$turquoise: #5da686;

// $border: #b0b0b0;
// $hover: #ececec;
// $even: #f6f6f6;

$primary: #34c6cd; // blue (primary)
$dprimary: #01939a; // dark primary
$gdprimary: #4faaae; // gray dark primary
$glprimary: #69b7ba; // gray light primary
$selectedprimary: #01939a;
$lightblue: #69b7ba;

$secondary: #f6cf7f; //brown
$dsecondary: #fbdda0; // beish

$frame: #9be0e4;
$error: #EE6841;
```

```
// $positive: #9cce00;
// $warning: #f9cf17;

/*
============================================================================
  Mixins

============================================================================
*/

@mixin linear-gradient($value...){
  background: -moz-linear-gradient(left, $value); /* FF3.6+ */
  background: -webkit-linear-gradient(left, $value); /* Chrome10+,Safari5.1+ */
  background: -o-linear-gradient(left, $value); /* Opera 11.10+ */
  background: -ms-linear-gradient(left, $value); /* IE10+ */
  // background: linear-gradient($value); /* W3C */
}

@mixin border-radius($radius) {
  -webkit-border-radius: $radius;
     -moz-border-radius: $radius;
      -ms-border-radius: $radius;
          border-radius: $radius;
}

@mixin box-shadow($top, $left, $blur, $color) {
  -webkit-box-shadow: $top $left $blur $color;
  -moz-box-shadow: $top $left $blur $color;
  box-shadow: $top $left $blur $color;
}

@mixin inset-shadow($length) {
  -moz-box-shadow:    inset 0 0 $length #000000;
  -webkit-box-shadow: inset 0 0 $length #000000;
  box-shadow:         inset 0 0 $length #000000;
}

@function black($opacity){
  @return rgba(0,0,0,$opacity)
}
@function white($opacity){
  @return rgba(255,255,255,$opacity)
}

@mixin box-emboss($opacity, $opacity2){
  box-shadow: white($opacity) 0 3px 0, inset black($opacity2) 0 3px 0;
}
```

8.1.1.2.2. application.css.scss

```
/*
 * This is a manifest file that'll be compiled into application.css, which will include all the files
 * listed below.
 *
 * Any CSS and SCSS file within this directory, lib/assets/stylesheets, vendor/assets/stylesheets,
 * or vendor/assets/stylesheets of plugins, if any, can be referenced here using a relative path.
 *
 * You're free to add application-wide styles to this file and they'll appear at the top of the
 * compiled file, but it's generally better to create a new file per style scope.
 *
 *= require_self
 *= require bootstrap
 *= require main
 *= require select2
 *= require select2-bootstrap
 *= require bootstrap-tagsinput
 *= require_tree .
 */
```

8.1.1.2.3. fonticons.css

```
@charset "UTF-8";

@font-face {
  font-family: "socialfeed";
  src:url("fonts/socialfeed.eot");
  src:url("fonts/socialfeed.eot?#iefix") format("embedded-opentype"),
    url("fonts/socialfeed.woff") format("woff"),
    url("fonts/socialfeed.ttf") format("truetype"),
    url("fonts/socialfeed.svg#socialfeed") format("svg");
  font-weight: normal;
  font-style: normal;

}

[data-icon]:before {
  font-family: "socialfeed" !important;
  content: attr(data-icon);
  font-style: normal !important;
  font-weight: normal !important;
  font-variant: normal !important;
  text-transform: none !important;
  speak: none;
  line-height: 1;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}
```

```
[class^="icon-"]:before,
[class*=" icon-"]:before {
  font-family: "socialfeed" !important;
  font-style: normal !important;
  font-weight: normal !important;
  font-variant: normal !important;
  text-transform: none !important;
  speak: none;
  line-height: 1;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

.icon-caret:before {
  content: "b";
}
.icon-arrow:before {
  content: "a";
}
.icon-dot:before {
  content: "d";
}
.icon-minus:before {
  content: "e";
}
.icon-plant:before {
  content: "f";
}
.icon-plus:before {
  content: "g";
}
.icon-refresh:before {
  content: "h";
}
.icon-tick:before {
  content: "j";
}
.icon-wave:before {
  content: "k";
}
.icon-star:before {
  content: "c";
}
.icon-caret-1:before {
  content: "i";
}
```

8.1.1.2.4. main.css.scss

```
@import "sass_defaults";

/*
================================================================================
  Base

================================================================================
*/

body{
  box-sizing: border-box;
}

.icon {
  color: $icongray;
  font-family: 'socialfeed';
  font-size: 20px;
}

h1{
  font-size: 42px;
}

.not-active {
  pointer-events: none;
  cursor: default;
  div{
    background-color: lighten($gray, 10%) !important;
    .entry &.icon{
      background-color: transparent !important;
      color: $gray !important;
    }
  }
}

.tooltip {
  z-index: 9999;
  position: fixed;
  word-wrap: break-word;
}

p, .bootstrap-tagsinput{
  font-size: 16px;
  input{
    font-size: 14px;
  }
}

.no-padding{
```

```
  padding-left: 0 !important;
  padding-right: 0 !important;
}

.no-margin{
  margin-left: 0 !important;
  margin-right: 0 !important;
}

.top-spacing-1{
  margin-top: 4em;
}

.top-spacing-2{
  margin-top: 2em;
}

.padding-bottom-1{
  padding-bottom: 0.7em;
}

/*
================================================================================
   Tables

================================================================================
*/
.table-container{
  overflow-y: auto;
  max-height: 220px;
  table.table{
    background-color: #caaa68;
    margin-bottom: 0;
    tr td{
      padding-top: 10px;
      padding-bottom: 6px;
      font-size: 16px;
      border: 1px solid;
      border-color: $secondary;
      &:last-of-type{
        padding-top: 7px;
        width: 10px;
      }
    }
    .icon{
      font-size: 12px;
      height: 24px;
      width: 24px;
      padding: 3px;
```

```
    }
  }
}

/*
================================================================================
  Forms

================================================================================
*/
.landing{
  form.form-inline{
    .form-group{
      display: inline;
      .paddings-sm{
        @media screen and (min-width: 480px) {
          padding: 0 20px;
        }
      }
      div[class^="col-"]{
        margin-top: 20px;
      }
      a{
        margin-left: 10px;
        color: $black;
        &:hover{
          background-color: transparent;
        }
      }
      label{
        font-weight: normal;
      }
      .field_with_errors{
        input{
          border-color: $error;
        }
        @include border-radius(5px);
        background-color: $error;
      }
    }
    .form-actions .button{
      margin-top: 20px;
    }
    input{
      background-color: rgba(#e6e6e6, 0.768);
      &.checkbox{
        width: 35px;
        height: 35px;
        margin-right: 10px;
```

```
      }
      &.btn{
        font-size: 16px;
        &.btn-warning{
          width: 120px;
          color: $dprimary;
          background-color: $lyellow;
          &:hover{
            background-color: lighten($lyellow, 5%);
          }
        }
        &.btn-primary{
          width: 80px;
          color: $white;
          background-color: $primary;
          &:hover{
            background-color: lighten($primary, 5%);
          }
        }
      }
      border-color: $primary;
      width: 100%;
    }
  }
}

.main-container{
  .dashboard .select2-container{
    margin-top: 5px;
    margin-bottom: -5px;
  }
  form{
    // TODO: Fix select styles when in dropdown state
    .form-control.select2-container{
      border: none;
      @include border-radius(12.5px);
      .select2-choices, .select2-choice{
        @include border-radius(12.5px);
      }
      .select2-container-active .select2-choice{
        @include border-radius(12.5px 12.5px 0 0);
      }
      .select2-drop-active{
        @include border-radius(0 0 12.5px 12.5px);
      }
    }
    .select2-container:not(.select2-container-active){
      .select2-choices, .select2-results{
        max-height: 10px;
```

```
    }
   }
   &.top-margin{
    margin-top: 5px;
   }
   &.bottom-margin{
    margin-bottom: 5px;
   }
   .input{
    margin-bottom: 22.5px;
    padding-left: 5px;
    &.field_with_errors{
     @include border-radius(15px);
     .error{
      white-space: nowrap;
      margin-right: 5px;
      float: right;
      width: 100%;
      text-align: right;
     }
    }
    label{
     vertical-align: top;
     margin-bottom: 0;
     padding-top: 8px;
    }
    label, button{
     font-size: 16px;
    }
    .bootstrap-tagsinput input{
     float: none;
    }
    input, select, textarea, .bootstrap-tagsinput{
     @include border-radius(12.5px);
     border: none;
     display: inline-block;
     width: 77%;
     float: right;
    }
   }
  }
}
/*
================================================================================
  Modals

================================================================================
*/
.modal-backdrop.in{
```

```
  opacity: 0;
}
.modal{
  &#notification{
    font-size: 16px;
    .danger{
      background-color: $error;
    }
    .info{
      background-color: $primary;
    }
  }
  .modal-content, .modal-body{
    .btn{
      color: $black !important;
      &:last-of-type{
        margin-left: 20px;
      }
      @include border-radius(10px);
    }
    @include border-radius(15px);
    padding: 10px;
    background-color: $dprimary;
    .panel{
      @include border-radius(15px);
      padding: 20px 20px 20px 15px;
      background-color: $primary;
    }
  }
}
/*
================================================================================
   Landing

================================================================================
*/
%layout-common{
  margin-left: -15px;
  margin-right: -15px;
  font-family: $font;
  overflow-x: hidden;
  width: 101.2%;

  .container-fluid{
    padding-right: 0;
  }
}

.landing{
```

```
  @extend %layout-common;
  @include linear-gradient(lighten($primary, 10%), $dprimary);
  .brand{
    margin-top: 60px;
    img{
      width: 350px;
      margin: 0 auto;
    }
    p{
      margin: 20px auto;
      color: $white;
    }
  }
  .box{
    padding: 20px 40px 40px;
    background-color: rgba($white, 0.58);
    background-image: url("rssbg-upper.svg");
    background-repeat: no-repeat;
    background-position: center bottom;
    @include border-radius(15px);
  }
  .rss-lower{
    margin: 0 auto 40px;
  }
  .frame{
    @include border-radius(15px);
    padding: 20px;
    background-color: $frame;
    width: 250px;
    height: 250px;
    &.center{
      margin: 0 auto 40px;
    }
    .text-center{
      line-height: 1.1;
      font-size: 18px;
    }
    img{
      margin: 0 auto 10px;
      height: 170px;
    }
  }
}

/*
================================================================================
  Layout
```

```
==========================================================================
*/
.main-container {
  @extend %layout-common;

  #header{
    @include linear-gradient($primary, $dprimary);
    color: $white;
    .logo{
      background-color: $dprimary;
    }
    .brand{
      margin-top: 60px;
      img{
        height: 56px;
      }
      p{
        margin-top: 10px;
        width: 340px;
      }
    }
    .icon{
      color: $white;
      font-size: 10px;
    }
  }

  #aside {
    padding: 4px;
    background-color: $primary;
    .box{
      @include border-radius(5px);
      margin-bottom: 4px;
      padding: 4px;
      background-color: $gdprimary;
      &.last{
        margin-bottom: 0;
      }
      .sidebar-btn{
        .btn{
          background-color: $glprimary;
          border: none;
          padding: 7px 12px;
        }
        &.selected .btn{
          background-color: $dprimary;
          color: darken($icongray, 10%);
        }
```

```
}
li a{
  padding: 15px;
  font-size: $fs-h4;
  color: $black;
  background-color: $glprimary;
  margin-bottom: 4px;
  &:hover{
    background-color: lighten($glprimary, 5%);
  }
  &:active{
    background-color: $selectedprimary;
  }
  &.selected{
    background-color: $dprimary;
  }
}
li:last-of-type a{
  margin-bottom: 0;
}
.sidebar-btn{
  &:hover{
    background-color: lighten($glprimary, 5%);
  }
  &:active{
    background-color: $selectedprimary;
  }
  &.selected{
    background-color: $dprimary;
  }
}
.title{
  margin: 10px 0;
  font-size: $fs-h4;
  color: $black;
}
.follow-panel{
  @include border-radius(5px);
  background-color: $lightblue;
  height: 30em;
  .error-message{
    padding: 5px;
  }
  .user{
    @include border-radius(5px);
    margin: 0.5em 3em 0.5em 0.5em;
    font-size: $fs-h4;
    padding: 0.4em 0.4em 0.4em 0;
    background-color: lighten($lightblue, 5%);
```

```
        width: 90%;
        &.selected{
          background-color: $dprimary;
          color: darken($icongray, 10%);
        }
        .btn{
          background-color: lighten($lightblue, 10%);
        }
        a, a:visited{
          color: $black;
        }
        .name{
          margin-top: 10px;
        }
      }
    }
  }
}
#content {
  margin-left: -30px;
  height: 48.8em;
  .dashboard{
    background-color: $dprimary;
    height: 66px;
    overflow: visible;
    width: 100%;
    padding: 15px;
    .btn{
      @include border-radius(10px);
      color: $gray;
      background-color: $white;
      border: none;
      padding: 3.5px 8px;
    }
  }
  .entries{
    @include inset-shadow(5px);
    padding: 15px;
    height: 43.8em;
    overflow-y: scroll;
    .entry{
      @include box-shadow(4px, 4px, 10px, lighten($gray, 10%));
      @include border-radius(10px);
      border: none;
      width: 100%;
      padding: 10px;
      margin-bottom: 15px;
      overflow-x: hidden;
      background-color: $secondary;
```

```
    img{
      border-radius: 5px;
      border: 2px solid $gray;
    }
    &.read{
      background-color: lighten($secondary, 10%);
    }
    .panel-title .icon{
      text-shadow: -1px 0 $turquoise, 0 1px $turquoise, 1px 0 $turquoise, 0 -1px $turquoise;
      color: $white;
      &.favorited{
        color: $yellow;
      }
    }
    .summary{
      padding-bottom: 7px;
    }
    .panel-heading, .panel-body{
      color: inherit;
      background-color: inherit;
      border: none;
    }
  }
  .error-message{
    color: $dprimary;
    margin-top: 40px;
    font-size: 30px;
  }
}
.secondary-view{
  width: 100%;
  height: 100%;
  padding: 20px;
  .panel{
    @include box-shadow(5px, 5px, 10px, $gray);
    @include border-radius(15px);
    width: 100%;
    height: 100%;
    padding: 20px;
    overflow: auto;
    background-color: $dsecondary;
    p{
      font-size: 14px;
    }
    .box{
      @include border-radius(15px);
      width: 100%;
      height: auto;
      margin-bottom: 20px;
```

```
padding: 10px 20px;
background-color: $secondary;
&:last-of-type{
  padding: 10px 20px 20px 20px;
}
h3{
  margin-top: 8px;
}
.description{
  margin-bottom: 65px;
  textarea{
    width: 73%;
  }
}
.top-margin{
  margin-top: 13px;
}
.bootstrap-tagsinput{
  width: 69%;
}
.keywords{
  margin-bottom: 5px;
  label{
    margin-right: 10px;
  }
}
.checkbox-group {
  margin-top: 10px;
  margin-bottom: 20px;
  label {
    font-size: 18px;
    font-weight: normal;
    float: left;
  }
  input {
    height: 35px;
    width: 35px;
    float: right;
  }
}
.filter-type, .filter-active{
  margin-bottom: 5px;
  float: right;
  label{
    font-size: 20px;
    display: inline-block;
  }
  .radio{
    margin-top: 0;
```

```
    display: inline-block;
    input{
      margin-top: 3px;
    }
  }
  input{
    margin-left: 20px;
    width: 35px;
    height: 35px;
  }
}
.tags{
  width: 85%;
  display: inline-block;
  margin-bottom: 15px;
  input{
    width: 77%;
  }
}
.btn{
  display: inline-block;
  float: right;
  background-color: $white;
  color: $gray;
  @include border-radius(10px);
}
.subpanel{
  @include border-radius(5px);
  min-height: 50px;
  margin-top: 10px;
  background-color: $dsecondary;
  padding: 15px;
  .title{
    font-size: 16px;
    font-weight: bold;
  }
  .delete-btn{
    background-color: lighten($dsecondary, 5%);
  }
  .icon{
    color: $gray;
    &.small{
      font-size: 12px;
    }
  }
  .feeds{
    margin-left: 15px;
  }
}
```

```
      }
    }
  }
 }
}

#footer{
  position: relative;
  background-color: $dprimary;
  height: 300px;
  .links{
    margin-top: 35px;
    list-style: none;
    line-height: 2;
    font-size: $fs-h4;
    a{
      color: $white;
    }
  }
  .content-decoration{
    position: absolute;
    bottom: 0px;
    height: 74px;
    width: 100%;
    background-repeat: no-repeat;
    background-image: url("plant.svg");
  }
  .footer-decoration{
    position: absolute;
    bottom: 0px;
    height: 65px;
    width: 100%;
    background-image: url("wave.svg");
  }
  .license{
    position: absolute;
    bottom: 5px;
    left: 20px;
    font-size: 14px;
    font-weight: bold;
  }
}
```

8.1.1.2.5. scaffolds.css.scss

```
@import "sass_defaults";

body {
  background-color: #fff;
```

```
  color: #333;
  font-family: verdana, arial, helvetica, sans-serif;
  font-size: 13px;
  line-height: 18px;
}

p, ol, ul, td {
  font-family: verdana, arial, helvetica, sans-serif;
  font-size: 13px;
  line-height: 18px;
}

pre {
  background-color: #eee;
  padding: 10px;
  font-size: 11px;
}

a {
  color: #000;
  &:visited {
    color: #666;
  }
  &:hover {
    color: #fff;
    background-color: #000;
  }
}

div {
  &.field, &.actions {
    margin-bottom: 10px;
  }
}

#notice {
  color: green;
}

.field_with_errors {
  padding: 2px;
  background-color: $error;
  display: table;
}

#error_explanation {
  width: 450px;
  border: 2px solid red;
  padding: 7px;
```

```
  padding-bottom: 0;
  margin-bottom: 20px;
  background-color: #f0f0f0;
  h2 {
    text-align: left;
    font-weight: bold;
    padding: 5px 5px 5px 15px;
    font-size: 12px;
    margin: -7px;
    margin-bottom: 0px;
    background-color: #c00;
    color: #fff;
  }
  ul li {
    font-size: 12px;
    list-style: square;
  }
}
```

8.1.2. controllers/

8.1.2.1. users/

8.1.2.1.1. registrations_controller.rb

```
class Users::RegistrationsController < Devise::RegistrationsController
  def update
    User.update current_user.id, params[:user]
    respond_to do |format|
      format.js { render 'notifications/refresh_user_notifications_form', status: :ok}
    end
  end
end
```

8.1.2.2. application_controller.rb

```
class ApplicationController < ActionController::Base
  protect_from_forgery
  set_current_tenant_through_filter
  before_filter :setup
  layout :layout_by_resource
  before_filter :authenticate_user!
  rescue_from SQLite3::BusyException, with: :database_locked

  def find_current_tenant
    set_current_tenant(current_user)
  end

  def set_followings
```

```
    @followings = FollowingDecorator.decorate_collection(Following.all)
  end

  def setup
    find_current_tenant
    set_followings
  end

  def database_locked
    redirect_to :back, alert: I18n.t('errors.messages.could_not_process')
  end

  protected
    def layout_by_resource
      if devise_controller?
        "landing"
      else
        "application"
      end
    end
end
```

8.1.2.3. categories_controller.rb

```
class CategoriesController < ApplicationController

  def index
    @categories = Category.order('name desc').all
    @category = Category.new
  end

  def show
    @categories = [Category.find(params[:id])]
    @category = Category.new
    respond_to do |format|
      format.html { render 'index', status: :ok }
    end
  end

  def new
    @category = Category.new
    respond_to do |format|
      format.js { render 'refresh_form', status: :ok }
    end
  end

  def edit
    @category = CategoriesService.edit params
    respond_to do |format|
```

```
    if @category.errors.empty?
      format.js { render 'refresh_form', status: :ok }
    else
      @error = @category.errors.full_messages.first
      format.js { render 'error', status: :ok }
    end
  end
end

def update
  @category = CategoriesService.update params
  respond_to do |format|
    if @category.errors.empty?
      @categories = Category.all
      @category = Category.new
      format.js { render 'refresh_all', status: :ok }
    else
      format.js { render 'new', status: :ok }
    end
  end
end

def create
  @category = CategoriesService.create params
  respond_to do |format|
    if @category.errors.empty?
      @categories = Category.all
      @category = Category.new
      format.js { render 'refresh_all', status: :ok }
    else
      format.js { render 'new', status: :ok }
    end
  end
end

def destroy
  category = CategoriesService.destroy params
  respond_to do |format|
    if category.errors.empty?
      @categories = Category.all
      format.js { render 'refresh_list', status: :ok }
    else
      @error = category.errors.full_messages.first
      format.js { render 'error', status: :unprocessable_entity }
    end
  end
end
end
```

8.1.2.4. entries_controller.rb

```
class EntriesController < ApplicationController

  def index
    if params[:user_id]
      @user = current_user.followeds.find(params[:user_id])
      ActsAsTenant.current_tenant = @user
    end
    @entries, @q = EntriesService.current_entries 50, params
    @entries = EntryDecorator.decorate_collection @entries
    @feed = Feed.new
    unless params[:category].blank?
      category_ids = params[:category].split
      @feeds = Feed.where(category_id: category_ids)
      @category = [Category.find(category_ids)].flatten
    else
      @feeds = Feed.all
    end
  end

  def show
    @entry = Entry.find params[:id]
  end

  def update
          @entry = EntriesService.update current_user, params
    @read_changed = true unless params[:entry][:read].blank?
    @entry = EntryDecorator.decorate @entry
          respond_to do |format|
      format.js
          end
  end
end
```

8.1.2.5. feeds_controller.rb

```
class FeedsController < ApplicationController

  def index
   @feeds = Feed.all
  end

  def show
    @feed = Feed.find params[:id]
  end

  def new
    @feed = Feed.new
```

```
    respond_to do |format|
      format.js
    end
  end

  def create
    @feed = FeedsService.create current_user, params
    respond_to do |format|
      if @feed.errors.empty?
        @entries, @q = EntriesService.current_entries 50
        @entries = EntryDecorator.decorate_collection @entries
        @feeds = Feed.all
        format.js
      else
        format.js{ render 'new', status: :unprocessable_entity }
      end
    end
  end

  def destroy
    feed = FeedsService.destroy params
    respond_to do |format|
      if feed.errors.empty?
        @categories = Category.all
        format.js { render 'categories/refresh_list', status: :ok }
      else
        format.js { render 'destroy_error', status: :unprocessable_entity }
      end
    end
  end
end
```

8.1.2.6. filters_controller.rb

```
class FiltersController < ApplicationController

  def index
    @feeds = Feed.all
    if params[:id]
      @feed = Feed.find params[:id]
      if @feed.filter
        @filter = @feed.filter
      else
        @filter = Filter.new feed_id: @feed.id
      end
    end
    respond_to do |format|
      format.html
      format.js { render 'refresh_form', status: :ok }
```

```
    end
  end

  def destroy
    @filter = FiltersService.destroy params
    respond_to do |format|
      if @filter.errors.empty?
        @feeds = Feed.all
        @filter = nil
        format.js { render 'refresh_all', status: :ok }
      else
        format.js { render 'refresh_form', status: :unprocessable_entity }
      end
    end
  end

  def update
    @filter = FiltersService.update params
    respond_to do |format|
      if @filter.errors.empty?
        format.js
      else
        format.js{ render 'refresh_form', status: :unprocessable_entity }
      end
    end
  end

  def create
    @filter = FiltersService.create params
    respond_to do |format|
      if @filter.errors.empty?
        format.js
      else
        format.js{ render 'refresh_form', status: :unprocessable_entity }
      end
    end
  end
end
```

8.1.2.7. followings_controller.rb

```
class FollowingsController < ApplicationController
  def index
    @users, @q = FollowingsService.search_users current_user, 50, params
    respond_to do |format|
      format.html
      format.js
    end
  end
```

```
  def create
    @following = FollowingsService.build current_user, params
    respond_to do |format|
      if @following.save
        format.js
      else
        format.js { render 'error', status: :ok }
      end
    end
  end

  def destroy
    @following = FollowingsService.destroy params
    respond_to do |format|
      if @following.errors.empty?
        format.js
      else
        format.js { render 'error', status: :ok }
      end
    end
  end
end
```

8.1.2.8. notifications_controller.rb

```
class NotificationsController < ApplicationController

  def index
    @feeds = Feed.all
    @user = current_user
    if params[:id]
      @feed = Feed.find params[:id]
      if @feed.notification
        @notification = @feed.notification
      else
        @notification = Notification.new feed_id: @feed.id
      end
    end
    respond_to do |format|
      format.html
      format.js { render 'refresh_form', status: :ok }
    end
  end

  def update
    @notification = NotificationsService.update params
    respond_to do |format|
      if @notification.errors.empty?
```

```
      format.js
    else
      format.js{ render 'refresh_form', status: :unprocessable_entity }
    end
  end
 end

 def destroy
  @notification = NotificationsService.destroy params
  respond_to do |format|
    if @notification.errors.empty?
      @feeds = Feed.all
      @notification = nil
      format.js { render 'refresh_all', status: :ok }
    else
      format.js { render 'refresh_form', status: :unprocessable_entity }
    end
  end
 end

 def create
  @notification = NotificationsService.create params
  respond_to do |format|
    if @notification.errors.empty?
      format.js
    else
      format.js{ render 'refresh_form', status: :unprocessable_entity }
    end
  end
 end
end
```

8.1.3. decorators/

8.1.3.1. category_decorator.rb

```
class CategoryDecorator < Draper::Decorator
 delegate_all

 # Define presentation-specific methods here. Helpers are accessed through
 # `helpers` (aka `h`). You can override attributes, for example:
 #
 #   def created_at
 #     helpers.content_tag :span, class: 'time' do
 #       object.created_at.strftime("%a %m/%d/%y")
 #     end
 #   end

 def to_s
```

```
      object.name
  end
end
```

8.1.3.2. entry_decorator.rb

```
class EntryDecorator < Draper::Decorator
  delegate_all

  decorates_association :feed
  # Define presentation-specific methods here. Helpers are accessed through
  # `helpers` (aka `h`). You can override attributes, for example:
  #
  #   def created_at
  #     helpers.content_tag :span, class: 'time' do
  #       object.created_at.strftime("%a %m/%d/%y")
  #     end
  #   end

  def id
    object.id
  end

  def read
    object.read ? Entry.human_attribute_name(:read) : Entry.human_attribute_name(:unread)
  end
end
```

8.1.3.3. feed_decorator.rb

```
class FeedDecorator < Draper::Decorator
  delegate_all

  decorates_association :category
  # Define presentation-specific methods here. Helpers are accessed through
  # `helpers` (aka `h`). You can override attributes, for example:
  #
  #   def created_at
  #     helpers.content_tag :span, class: 'time' do
  #       object.created_at.strftime("%a %m/%d/%y")
  #     end
  #   end

  def to_s
    object.name || I18n.t('placeholders.untitled')
  end
end
```

8.1.3.4. following_decorator.rb

```
class FollowingDecorator < Draper::Decorator
  delegate_all

  # Define presentation-specific methods here. Helpers are accessed through
  # `helpers` (aka `h`). You can override attributes, for example:
  #
  #   def created_at
  #     helpers.content_tag :span, class: 'time' do
  #       object.created_at.strftime("%a %m/%d/%y")
  #     end
  #   end

  def followed
    object.followed.name
  end
end
```

8.1.4. helpers/

8.1.4.1. application_helper.rb

```
module ApplicationHelper
  def sidebar_link_to name, controller_name, &block
    if block_given?
      link_to send("#{name}_path"), class: "text-center sidebar-btn #{'selected' if controller_name ==
name}", &block
    else
      link_to send("#{name}_path"), class: "text-center #{'selected' if controller_name == name}" do
        I18n.t("layouts.aside.#{name}")
      end
    end
  end
end
```

8.1.4.2. categories_helper.rb

```
module CategoriesHelper
  def category_title category
    title = ''
    title += I18n.t('categories.index.description') + ": #{category.description}<br>" unless
category.description.blank?
    title += I18n.t('categories.index.tags') + ": #{category.tags}" unless category.tags.blank?
    title.chomp
  end
end
```

8.1.4.3. entries_helper.rb

```
module EntriesHelper
```

```
        def read_link entry
                link_to entry.read, entry_path(entry, params: {entry: {read: !entry.object.read}}),
method: :put, :remote => true, class: "#{'not-active' if @user}"
        end

        def favorite_link entry
                link_to entry_path(entry, params: {entry: {favorite: !entry.object.favorite}}),
method: :put, :remote => true, class: "#{'not-active' if @user}" do
                        content_tag(:div, 'c', class: "icon pull-right #{'favorited' if entry.favorite?}")
                end
        end

        def read? entry
                'read' if entry.object.read?
        end

        def truncated_name_with_tooltip name, length
                if name.size > length
                        content_tag(:span, truncate(name, length: length), title: name, data:
{toggle: "tooltip", placement: "bottom"})
                else
                        content_tag(:span, name)
                end
        end
end
```

8.1.4.4. followings_helper.rb

```
module FollowingsHelper
  def toggle_follow_button follower, followed
    if follower.followed_ids.include? followed.id
      link_to following_path(id: follower.followings.select {|f| f.followed_id == followed.id}.first.id),
method: :delete, remote: true do
        content_tag(:div, 'e', class: "btn icon")
      end
    else
      link_to followings_path(followed_id: followed.id), method: :post, remote: true do
        content_tag(:div, 'g', class: "btn icon")
      end
    end
  end
end
```

8.1.5. mailers/

8.1.5.1. notification_mailer.rb

```
class NotificationMailer < ActionMailer::Base
  add_template_helper(EntriesHelper)
```

```
  default from: "from@example.com"

  def notify_entries user, notifications
    @user = user
    @notifications = notifications
    mail(to: @user.email, subject:
I18n.t('notification_mailer.notify_entries.important_topic_notification'))
  end

  def notify_following follower, followed
    @follower = follower
    @followed = followed
    mail(to: @followed.email, subject:
I18n.t('notification_mailer.notify_entries.following_notification'))
  end
end
```

8.1.6. models/

8.1.6.1. category.rb

```
class Category < ActiveRecord::Base
  acts_as_tenant :user
  before_destroy :check_if_destroyable
  before_update :check_if_editable
  #
  # schema
  #
  fields do
    name        :string
    tags        :string
    description :text
    editable    :boolean, default: true
    timestamps
  end

  attr_accessible :name, :tags, :description, :editable, :user_id

  #
  # Validations
  #
  validates :name, presence: true
  validates_uniqueness_to_tenant :name
  #
  # Scopes
  #
  scope :undefined, -> { where(editable: false) }

  #
```

```
  # Relations
  #
  has_many   :feeds, dependent: :destroy
  belongs_to :user
  #
  # Methods
  #
  protected
    def check_if_destroyable
      raise I18n.t('errors.can_not_be_destroyed') unless editable
    end
    def check_if_editable
      raise I18n.t('errors.can_not_be_edited') unless editable
    end
end
```

8.1.6.2. entry.rb

```
class Entry < ActiveRecord::Base
  acts_as_tenant :user
  #
  # schema
  #
  fields do
    name         :string
    summary      :text
    content      :text
    url          :string
    published_at :datetime
    guid         :string
    favorite     :boolean, default: false
    read         :boolean, default: false
    user_id      :integer
    timestamps
  end


  attr_accessible :name, :summary, :content, :url, :published_at, :guid, :feed_id, :read, :favorite,
:user_id

  #
  # Validations
  #

  #
  # Scope
  #
  scope :deletable, -> { where(favorite: false) }
```

```
  #
  # Relations
  #
  belongs_to :feed
  #
  # Methods
  #
  def self.add_entries entries, feed_id, user_id
    entries.each do |entry|
      unless exists? guid: entry.id
        create!(
          name:         entry.title,
          summary:      entry.summary,
          content:      entry._?.content,
          url:          entry.url,
          published_at: entry.published,
          guid:         entry.id,
          feed_id:      feed_id,
          user_id:      user_id
        )
      end
    end
  end

  def <=> other_entry
    other_entry.published_at <=> published_at
  end
end
```

8.1.6.3. feed.rb

```
require 'feedjira'

class Feed < ActiveRecord::Base
  acts_as_tenant :user
  before_validation :set_default_name, if: Proc.new { |feed| feed.name.blank? }
  #
  # schema
  #
  fields do
    name          :string
    url           :string
    description   :text
    etag          :string
    last_modified :datetime
    user_id       :integer
    timestamps
  end
```

```
attr_accessible :name, :url, :description, :category_id

#
# Validations
#
validates :url, presence: true, feed_url: true
validates :category_id, presence: true
validates :name, presence: true
validates_uniqueness_to_tenant [:name, :url]
#
# Relations
#
has_many   :entries, dependent: :destroy
has_one    :notification, dependent: :destroy
has_one    :filter, dependent: :destroy
belongs_to :category
#
# Methods
#

def get_latest_entries n
  entries.sort.first(n)
end

def update_entries
  feed = Feedjira::Feed.fetch_and_parse(url)
  raise 'Feed format issue' if feed.is_a? Numeric
  notification_entries = []
  unless self.etag == feed.etag && self.last_modified == feed.last_modified
    apply_filter!(feed) if filter._?.active
    user = User.find(user_id)
    if notification._?.active
      notification_entries = search_notification_entries(feed, user)
    end
    self.etag = feed.etag
    self.last_modified = feed.last_modified
    Entry.add_entries(feed.entries.first(50), self.id, user_id)
  end
  notification_entries
end

protected
  def set_default_name
    self.name = Feedjira::Feed.fetch_and_parse(url).title
  rescue Exception => e
  end

  # filter out entries
  def apply_filter! feed
```

```
    keywords = filter.keywords.split(',')
    entries = feed.entries.select do |entry|
      if filter.list_type # whitelisting
        entry_contains_any_keyword?(entry, keywords)
      else # blacklisting
        !entry_contains_any_keyword?(entry, keywords)
      end
    end
    feed.entries = entries
  end

  # run email notifications
  def search_notification_entries feed, user
    keywords = notification.keywords.split(',')
    notification_entries = []
    feed.entries.each do |entry|
      if !Entry.exists?(guid: entry.id) && entry_contains_any_keyword?(entry, keywords)
        notification_entries << entry
      end
    end
    notification_entries
  end

  # check if the entry contains any of the keywords
  def entry_contains_any_keyword? entry, keywords
    keywords.any? do |keyword|
      entry.title.include?(keyword) ||
      entry.summary.include?(keyword) ||
      entry._?.content._?.include?(keyword)
    end
  end
end
```

8.1.6.4. filter.rb

```
class Filter < ActiveRecord::Base
  acts_as_tenant :user
  #
  # schema
  #
  fields do
    keywords  :string
    list_type :boolean, default: false # true => whitelist, false => blacklist
    active    :boolean, default: true
    user_id   :integer
    timestamps
  end

  attr_accessible :keywords, :list_type, :active, :feed_id
```

```
  #
  # Validations
  #
  validates :user_id, presence: true
  validates :keywords, presence: true

  #
  # Relations
  #
  belongs_to :feed
  #
  # Methods
  #
end
```

8.1.6.5. following.rb

```
class Following < ActiveRecord::Base
  acts_as_tenant :user
  after_create :notify_following, if: Proc.new { followed.notify_new_follower }
  #
  # schema
  #
  fields do
    timestamps
  end

  attr_protected

  #
  # Validations
  #
  validates_uniqueness_to_tenant :followed_id

  #
  # Relations
  #
  belongs_to :user
  belongs_to :followed, class_name: 'User'

  #
  # Methods
  #
  private
    def notify_following
      NotificationMailer.notify_following(user, followed).deliver
    end
end
```

8.1.6.6. notification.rb

```
class Notification < ActiveRecord::Base
  acts_as_tenant :user
  #
  # schema
  #
  fields do
    keywords :string
    active   :boolean, default: true
    user_id  :integer
    timestamps
  end

  attr_accessible :keywords, :active, :feed_id

  #
  # Validations
  #
  validates :user_id, presence: true
  validates :keywords, presence: true

  #
  # Relations
  #
  belongs_to :feed
  #
  # Methods
  #
end
```

8.1.6.7. user.rb

```
class User < ActiveRecord::Base
  # Include default devise modules. Others available are:
  # :confirmable, :lockable, :timeoutable and :omniauthable
  devise :database_authenticatable, :registerable,
         :recoverable, :rememberable, :trackable, :validatable

  # Setup accessible (or protected) attributes for your model
  attr_accessible :name, :email, :notify_new_follower, :notify_site_updates,
:notify_important_topics, :password, :password_confirmation, :remember_me
  # attr_accessible :title, :body
  after_create :associate_undefined_category
  before_destroy :skip_category_callbacks
  #
  # schema
  #
```

```
  fields do
          name                                                :string
    notify_new_follower     :boolean, default: true
    notify_site_updates     :boolean, default: true
    notify_important_topics :boolean, default: true
          email                                               :string, default: "", null: false, index:
"index_users_on_email", unique: true
          encrypted_password                  :string, default: "", null: false
          reset_password_token      :string, index: "index_users_on_reset_password_token",
unique: true
          reset_password_sent_at  :datetime
          remember_created_at                 :datetime
          sign_in_count                         :integer, default: 0, null: false
          current_sign_in_at                  :datetime
          last_sign_in_at                       :datetime
          current_sign_in_ip                  :string
          last_sign_in_ip                       :string
          remember_created_at      :datetime
          created_at                                          :datetime, null: false
          updated_at                                          :datetime, null: false
  end

  #
  # Validations
  #
  validates :name, presence: true

  #
  # Relations
  #
  has_many :followings
  has_many :followeds, through: :followings
  has_many :inverse_followings, class_name: "Following", :foreign_key => "followed_id"
  has_many :inverse_followeds, through: :inverse_followings, :source => :user

  has_many :categories, dependent: :destroy
  has_many :feeds, through: :categories

  #
  # Scopes
  #
  scope :except, ->(user) { where('id != ?', user) }

  #
  # Methods
  #

  def skip_category_callbacks
    Category.skip_callback(:destroy, :before, :check_if_destroyable)
```

```
  end

  def associate_undefined_category
    Category.create! name: I18n.t('placeholders.undefined'), editable: false, user_id: self.id
  end
end
```

8.1.7. services/

8.1.7.1. categories_service.rb

```
class CategoriesService
  # Fetches and parses entries from all feeds of the user and returns them
  def self.create params
    Category.create params[:category]
  end

  def self.update params
    Category.update params[:id], params[:category]
  end

  def self.edit params
    category = Category.find params[:id]
    category.errors.add(:category, I18n.t('errors.can_not_be_edited')) unless category.editable
    category
  end

  def self.destroy params
    Category.destroy params[:id]
  rescue Exception => e
    category = Category.find params[:id]
    category.errors.add(:category, e.message)
    category
  end
end
```

8.1.7.2. entries_service.rb

```
class EntriesService
        # Fetches and parses entries from all feeds of the user and returns them
  def self.latest_entries user, quantity
    feeds = Feed.all
    begin
        feeds.each(&:update_entries)
    rescue Exception => e
    end
    entries = feeds.map{ |feed| feed.get_latest_entries(quantity) }
    entries.flatten.sort.slice 0, quantity
  end
```

```
  def self.update user, params
    Entry.update params[:id], params[:entry]
  end

  # Returns the database stored entries of all feeds from the user
  def self.current_entries quantity, search_params = nil
    q = search_params ? search_params[:q] : nil
    search = Entry.ransack(q)
    if search_params && search_params[:q].blank? && !search_params[:category].blank?
      categories = Category.find search_params[:category]
      entries = categories.map{|category| category.feeds.map {|feed| feed.entries }}
      entries.flatten!
      entries = entries.sort.slice(0..quantity)
    else
      entries = search.result(distinct: true).order('published_at desc').first(quantity)
    end
    [entries, search]
  end
end
```

8.1.7.3. feeds_service.rb

```
class FeedsService
  def self.create user, params
    feed = Feed.create params[:feed]
    if feed.errors.empty?
      feed.update_entries
    end
    feed
  end

  def self.destroy params
    Feed.destroy params[:id]
  end
end
```

8.1.7.4. filters_service.rb

```
class FiltersService
  # Fetches and parses entries from all feeds of the user and returns them
  def self.create params
    Filter.create params[:filter]
  end

  def self.update params
    Filter.update params[:id], params[:filter]
  end
```

```
  def self.edit params
    Filter.find params[:id]
  end

  def self.destroy params
    Filter.destroy params[:id]
  rescue Exception => e
    filter = Filter.find params[:id]
    filter.errors.add(:filter, e.message)
    filter
  end
end
```

8.1.7.5. followings_service.rb

```
class FollowingsService
  # Fetches and parses entries from all feeds of the user and returns them
  def self.search_users user, quantity, search_params
    search = User.except(user).ransack(search_params[:q])
    entries = search.result(distinct: true).order('email desc').first(quantity)
    [entries, search]
  end

  def self.build user, params
    user.followings.build(followed_id: params[:followed_id])
  end

  def self.destroy params
    Following.destroy params[:id]
  end
end
```

8.1.7.6. notifications_service.rb

```
class NotificationsService
  # Fetches and parses entries from all feeds of the user and returns them
  def self.create params
    Notification.create params[:notification]
  end

  def self.update params
    Notification.update params[:id], params[:notification]
  end

  def self.edit params
    Notification.find params[:id]
  end

  def self.destroy params
```

```
    Notification.destroy params[:id]
  rescue Exception => e
    notification = Notification.find params[:id]
    notification.errors.add(:notification, e.message)
    notification
  end
end
```

8.1.8. validators/

8.1.8.1. feed_url_validator.rb

```
require 'feedjira'

class FeedUrlValidator < ActiveModel::EachValidator
  def validate_each(record, attribute, value)
    feed = Feedjira::Feed.fetch_and_parse value
    if feed.is_a? Fixnum
      record.errors[attribute] << (options[:message] || I18n.t('errors.messages.invalid_feed_url'))
    end
  end
end
```

8.1.9. views/

8.1.9.1. categories/

8.1.9.1.1. _categories_list.html.haml

```
- categories.each do |category|
  = render 'categories/category', category: category
```

8.1.9.1.2. _category.html.haml

```
.subpanel
  .row
    .col-xs-1
      - if category.feeds.any?
        .icon{data: {toggle: "collapse", target: "#category-feeds#{category.id}"}, aria: {expanded:
"false", controls: "collapseExample"}}
          a
    .col-xs-10
      %span.title{data: {toggle: "tooltip", placement: "top"}, title: category_title(category)}
        - if category.editable
          = link_to category.name, edit_category_path(category), remote: true
        - else
          = category.name
    .col-xs-1
      - if category.editable
```

```
      = link_to category_path(category), method: :delete, remote: true, data: {confirm:
category.feeds.any? ? t('.are_you_sure') : nil} do
        .btn.icon.delete-btn
          e
  .feeds.row.collapse{id: "category-feeds#{category.id}"}
    - category.feeds.each do |feed|
      .col-xs-1
        = link_to feed_path(feed), method: :delete, remote: true do
          .icon.small
            e
      .col-xs-11
        %span{data: {toggle: "tooltip", placement: "right"}, title: feed.url}
          = feed.name
```

8.1.9.1.3. _category_form.html.haml

```
%h3
  .row
    - unless action_name == 'edit'
      .col-xs-12
        = t('.new')
    - else
      .col-xs-8
        = t('.edit')
      .col-xs-4
        = link_to t('.new'), new_category_path(), remote: true, class: 'btn'
= simple_form_for @category, remote: true do |f|
  = f.input :name, required: false, placeholder: t('.enter_category_name'), wrapper_html: {class:
'form-group'}, input_html: {class: 'form-control'}
  = f.input :description, placeholder: t('.enter_description'), wrapper_html: {class: 'form-group
description'}, input_html: {class: 'form-control', rows: 3}
  = f.input :tags, placeholder: t('.enter_tags'), wrapper_html: {class: 'form-group tags'}, input_html:
{class: 'form-control', data: {role: 'tagsinput'}}
  %button.btn.icon
    g
```

8.1.9.1.4. error.js.erb

```
$(document).ready(function(){
  alert('<%= @error %>');
});
```

8.1.9.1.5. index.html.haml

```
.secondary-view
  .panel
    .row
      .col-xs-8.col-xs-offset-2
        %h2
```

```
        = t('.categories')
      %p
        = t('.create_a_new_category')
      .box
        .js-category-form-container
          = render 'category_form', category: @category
      .box
        %h3
          = t('.list')
        .js-categories-list-container
          = render 'categories_list', categories: @categories

= content_for :bottom_js do
  = javascript_include_tag "categories/categories"
```

8.1.9.1.6. new.js.erb

```
$(document).ready(function(){
  var category = new Category();
  $category_form = $("<%= j render 'category_form', category: @category %>");
  category.refresh_form($category_form);
});
```

8.1.9.1.7. refresh_all.js.erb

```
$(document).ready(function(){
  var category = new Category();
  $category_form = $("<%= j render 'category_form', category: @category %>");
  category.refresh_form($category_form);

  $categories_list = $("<%= j render 'categories/categories_list', categories: @categories %>");
  category.refresh_list($categories_list);
});
```

8.1.9.1.8. refresh_form.js.erb

```
$(document).ready(function(){
  var category = new Category();
  $category_form = $("<%= j render 'category_form', category: @category %>");
  category.refresh_form($category_form);
});
```

8.1.9.1.9. refresh_list.js.erb

```
$(document).ready(function(){
  var category = new Category();
  $categories_list = $("<%= j render 'categories/categories_list', categories: @categories %>");
  category.refresh_list($categories_list);
});
```

8.1.9.2. devise/

8.1.9.2.1. confirmations/

8.1.9.2.1.1. new.html.haml

```
%h2 Resend confirmation instructions
= simple_form_for(resource, as: resource_name, url: confirmation_path(resource_name), html: {
method: :post }) do |f|
  = f.error_notification
  = f.full_error :confirmation_token
  .form-inputs
    = f.input :email, required: true, autofocus: true
  .form-actions
    = f.button :submit, "Resend confirmation instructions"
= render "devise/shared/links"
```

8.1.9.2.2. mailer/

8.1.9.2.2.1. confirmation_instructions.html.haml

```
%p
  Welcome #{@email}!
%p You can confirm your account email through the link below:
%p= link_to 'Confirm my account', confirmation_url(@resource, confirmation_token: @token)
```

8.1.9.2.2.2. reset_password_instructions.html.haml

```
%p
  Hello #{@resource.email}!
%p Someone has requested a link to change your password. You can do this through the link
below.
%p= link_to 'Change my password', edit_password_url(@resource, reset_password_token:
@token)
%p If you didn't request this, please ignore this email.
%p Your password won't change until you access the link above and create a new one.
```

8.1.9.2.2.3. unlock_instructions.html.haml

```
%p
  Hello #{@resource.email}!
%p Your account has been locked due to an excessive number of unsuccessful sign in attempts.
%p Click the link below to unlock your account:
%p= link_to 'Unlock my account', unlock_url(@resource, unlock_token: @token)
```

8.1.9.2.3. passwords/

8.1.9.2.3.1. edit.html.haml

```
%h2 Change your password
= simple_form_for(resource, as: resource_name, url: password_path(resource_name), html: {
method: :put }) do |f|
  = f.error_notification
  = f.input :reset_password_token, as: :hidden
  = f.full_error :reset_password_token
  .form-inputs
    = f.input :password, label: "New password", required: true, autofocus: true
    = f.input :password_confirmation, label: "Confirm your new password", required: true
  .form-actions
    = f.button :submit, "Change my password"
= render "devise/shared/links"
```

8.1.9.2.3.2. new.html.haml

```
%h2 Forgot your password?
= simple_form_for(resource, as: resource_name, url: password_path(resource_name), html: {
method: :post }) do |f|
  = f.error_notification
  .form-inputs
    = f.input :email, required: true, autofocus: true
  .form-actions
    = f.button :submit, "Send me reset password instructions"
= render "devise/shared/links"
```

8.1.9.2.4. registrations/

8.1.9.2.4.1. _new.html.haml

```
%h2
  = t('.sign_up')
= simple_form_for resource, as: resource_name, url: registration_path(resource_name), html:
{novalidate: true, class: 'form-inline'} do |f|
  .form-inputs.form-group
    .paddings-sm
      .row
        .col-md-6.col-xs-12
          = f.input :email, required: false, autofocus: controller_name == 'registrations', label: false,
placeholder: User.human_attribute_name(:email), input_html: { class: 'form-control'}
        .col-md-6.col-xs-12
          = f.input :name, required: false, label: false, placeholder: t('.pick_a_username'), input_html: {
class: 'form-control'}
      .row
        .col-md-6.col-xs-12
          = f.input :password, required: false, label: false, placeholder:
User.human_attribute_name(:password), input_html: { class: 'form-control'}
        .col-md-6.col-xs-12
```

```
        = f.input :password_confirmation, required: false, label: false, placeholder:
t('.reenter_password'), input_html: { class: 'form-control'}
  .form-actions.text-center
    = f.button :submit, "#{t('.sign_up')}!", class: 'btn btn-warning'
-# = render "devise/shared/links"
```

8.1.9.2.4.2. edit.html.haml

```
%h2
  Edit #{resource_name.to_s.humanize}
= simple_form_for(resource, as: resource_name, url: registration_path(resource_name), html: {
method: :put }) do |f|
  = f.error_notification
  .form-inputs
    = f.input :email, required: true, autofocus: true
    - if devise_mapping.confirmable? && resource.pending_reconfirmation?
      %p
        Currently waiting confirmation for: #{resource.unconfirmed_email}
    = f.input :password, autocomplete: "off", hint: "leave it blank if you don't want to change it",
required: false
    = f.input :password_confirmation, required: false
    = f.input :current_password, hint: "we need your current password to confirm your changes",
required: true
  .form-actions
    = f.button :submit, "Update"
%h3 Cancel my account
%p
  Unhappy? #{link_to "Cancel my account", registration_path(resource_name), data: { confirm:
"Are you sure?" }, method: :delete}
= link_to "Back", :back
```

8.1.9.2.5. sessions/

8.1.9.2.5.1. _new.html.haml

```
%h2
  = t('.already_got_an_account')
= simple_form_for resource, as: resource_name, url: session_path(resource_name), html:
{novalidate: true, class: 'form-inline' } do |f|
  .form-inputs.form-group
    .paddings-sm
      .row
        .col-md-6.col-sm-8.col-xs-12
          = f.input :email, required: false, autofocus: controller_name == 'sessions', label: false,
placeholder: User.human_attribute_name(:email), input_html: {class: 'form-control'}
        .col-md-6.col-sm-4.col-xs-12
          = f.input :remember_me, as: :boolean, label: t('.remember_me'), input_html: {class:
'form-control checkbox'} if devise_mapping.rememberable?
      .row
```

```
      .col-md-6.col-xs-12
        = f.input :password, required: false, label: false, placeholder:
User.human_attribute_name(:password), input_html: {class: 'form-control'}
      .col-md-6.col-xs-12
        = f.button :submit, t('.sign_in'), class: 'btn btn-default btn-primary'
        -# = render "devise/shared/links"
```

8.1.9.2.6. shared/

8.1.9.2.6.1. _links.html.haml

```
- if controller_name != 'sessions'
  = link_to "Log in", new_session_path(resource_name)
  %br/
-# - if devise_mapping.registerable? && controller_name != 'registrations'
-#   = link_to "Sign up", new_registration_path(resource_name)
  -# %br/
- if devise_mapping.recoverable? && controller_name != 'passwords' && controller_name !=
'registrations'
  = link_to t('.forgot_your_password'), new_password_path(resource_name)
  %br/
- if devise_mapping.confirmable? && controller_name != 'confirmations'
  = link_to "Didn't receive confirmation instructions?", new_confirmation_path(resource_name)
  %br/
- if devise_mapping.lockable? && resource_class.unlock_strategy_enabled?(:email) &&
controller_name != 'unlocks'
  = link_to "Didn't receive unlock instructions?", new_unlock_path(resource_name)
  %br/
- if devise_mapping.omniauthable?
  - resource_class.omniauth_providers.each do |provider|
    = link_to "Sign in with #{provider.to_s.titleize}", omniauth_authorize_path(resource_name,
provider)
    %br/
```

8.1.9.2.7. unlocks/

8.1.9.2.7.1. new.html.haml

```
%h2 Resend unlock instructions
= simple_form_for(resource, as: resource_name, url: unlock_path(resource_name), html: {
method: :post }) do |f|
  = f.error_notification
  = f.full_error :unlock_token
  .form-inputs
    = f.input :email, required: true, autofocus: true
  .form-actions
    = f.button :submit, "Resend unlock instructions"
= render "devise/shared/links"
```

8.1.9.3. entries/

8.1.9.3.1. _category_choice_form.html.haml

```
= simple_form_for entries_path, method: :get do |f|
  .col-xs-4.input
    = select_tag :category, options_from_collection_for_select(Category.all, :id, :name,
category._?.map(&:id)), multiple: true, placeholder: t('placeholders.all'), class: 'form-control'
    = hidden_field_tag :user_id, @user.id if @user
```

8.1.9.3.2. _entries.html.haml

```
- entries.each do |entry|
  = render 'entries/entry', entry: entry
```

8.1.9.3.3. _entries_search_form.html.haml

```
= search_form_for q, url: entries_path, builder: SimpleForm::FormBuilder do |f|
  .col-xs-3
    = f.input :feed_id_in, collection: feeds, required: false, include_blank: false, placeholder:
t('placeholders.all'), label: false, input_html: {class: 'form-control', multiple: true}
  .col-xs-2
    = f.input :read_eq, collection: [[t('.read'), true],[t('.unread'), false]], include_blank:
t('placeholders.all'), required: false, label: false, input_html: {class: 'form-control'}
  .col-xs-2
    = f.input :favorite_eq, collection: [[t('.favorite'), true],[t('.not_favorite'), false]], include_blank:
t('placeholders.all'), required: false, label: false, input_html: {class: 'form-control'}
  .col-xs-1
    = hidden_field_tag :category, category._?.map(&:id)
    = hidden_field_tag :user_id, @user.id if @user
    %button.btn.icon
      h
```

8.1.9.3.4. _entry.html.haml

```
.entry.js-entry.panel.panel-default{id: "id-#{entry.id}", class: read?(entry)}
  #headingOne.panel-heading{:role => "tab"}
    .panel-title
      %a.collapsed{"aria-controls" => "collapse#{entry.id}", "aria-expanded" => "true", "data-parent"
=> "#accordion", "data-toggle" => "collapse", :href => "#collapse#{entry.id}"}
        .row
          .col-xs-5
            .name.pull-left
              = truncated_name_with_tooltip entry.name, 45
          .col-xs-5
            %span.published-at
              = time_ago_in_words entry.published_at
            = '/'
            %span
```

```
          = link_to entry.feed.category, category_path(entry.feed.category), class: "#{'not-active' if
@user}"
        = '/'
        %a
          = entry.feed
      .col-xs-2
        %span.js-read-link
          = read_link(entry)
        %span.js-favorite-link
          = favorite_link(entry)
      .col-xs-12
  .panel-collapse.collapse{"aria-labelledby" => "headingOne", :role => "tabpanel", id:
"collapse#{entry.id}"}
    .panel-body.summary
      = raw entry.summary
      = raw entry.content
      %br
      = link_to t('.view_website'), entry.url, target: '_blank'
```

8.1.9.3.5. index.html.haml

```
.dashboard
  .row
    .col-xs-1
      = render 'feeds/modal_new_feed_form', feed: @feed
    .col-xs-11
      .row
        %span.js-category-form
          = render 'category_choice_form', category: @category
        %span.js-entries-search-form
          = render 'entries_search_form', q: @q, feeds: @feeds, category: @category
.entries.js-entries
  - if @entries.empty?
    .error-message.text-center
      .glyphicon.glyphicon-exclamation-sign{aria: {hidden: 'true'}}
      = t('.no_entries_found')
  - else
    #accordion.panel-group{"aria-multiselectable" => "true", :role => "tablist"}
      = render 'entries', entries: @entries

= content_for :bottom_js do
  = javascript_include_tag "entries/entries"
```

8.1.9.3.6. update.js.erb

```
$(document).ready(function(){
  var app = new App();
  $entry = $('.entry#id-<%= "#{@entry.id}" %>');
  <% if @read_changed %>
```

```
    $selector = $entry.find('.js-read-link');
    $link = $('<%= read_link(@entry) %>');
    app.refresh_content($selector, $link, function(){
      $entry.toggleClass('read');
    });
  <% else %>
    $selector = $entry.find('.js-favorite-link');
    $link = $('<%= favorite_link(@entry) %>');
    app.refresh_content($selector, $link);
  <% end %>
});
```

8.1.9.4. feeds/

8.1.9.4.1. _feed_form.html.haml

```
= simple_form_for feed, remote: true do |f|
  .modal-body
    .row
      .col-xs-12
        .panel
          = render 'form_fields', f: f
          .clearfix
    .row
      .col-xs-5.col-xs-offset-7
        .pull-right
          %button.btn{type: "button", data: {dismiss: "modal"}}
            %span.text
              = t('.cancel')
          %button.btn.js-submit{type: "button", data: { disable_with: t('.adding') }}
            %span.text
              = t('.add')
```

8.1.9.4.2. _form_fields.html.haml

```
= f.input :url, required: false, placeholder: t('.paste_url_here'), wrapper_html: {class: 'form-group'},
input_html: {class: 'form-control'}
= f.association :category, as: :select, required: false, selected: Category.undefined, wrapper_html:
{class: 'form-group'}, input_html: {class: 'form-control'}
= f.input :name, required: false, placeholder: t('.give_custom_title'), wrapper_html: {class:
'form-group'}, input_html: {class: 'form-control'}
= f.input :description, placeholder: t('.add_description'), wrapper_html: {class: 'form-group'},
input_html: {class: 'form-control', rows: 4}
```

8.1.9.4.3. _modal_new_feed_form.html.haml

```
/ Button trigger modal
= link_to new_feed_path, remote: true, class: "#{'not-active' if @user}" do
  .btn.icon
```

```
  g

/ Modal
#add-feed-modal.modal.fade{"aria-hidden" => "true", "aria-labelledby" => "myModalLabel", :role =>
"dialog", :tabindex => "-1"}
  .modal-dialog
    .modal-content.js-form-container
```

8.1.9.4.4. create.js.erb

```
$(document).ready(function(){
  var entry = new Entry();
  $entries_form = $("<%= j render 'entries/entries_search_form', q: @q, feeds: @feeds, category:
nil %>");
  entry.refresh_search_form($entries_form);
  $entries = $("<%= j render 'entries/entries', entries: @entries %>");
  entry.refresh_entries($entries);
});
```

8.1.9.4.5. destroy_error.js.erb

```
$(document).ready(function(){
  alert('Could not delete feed'); //TODO: Apply I18n and use Bootstrap alert
});
```

8.1.9.4.6. new.js.erb

```
$(document).ready(function(){
  var feed = new Feed();
  $form = $("<%= j render 'feed_form', feed: @feed %>");
  feed.refresh_form($form);
});
```

8.1.9.5. filters/

8.1.9.5.1. _feed_selection.html.haml

```
.row
  = form_tag filters_path, method: :get, remote: true do |f|
    .col-xs-12
      = select_tag :id, options_from_collection_for_select(feeds, :id, :name, feed._?.id),
include_blank: true, class: 'form-control', data: {placeholder: t('placeholders.select_feed')}
```

8.1.9.5.2. _filter_form.html.haml

```
- if filter
  .box
    = simple_form_for filter, remote: true do |f|
      .row
```

```
        .col-xs-12
          .pull-right
            = f.input :list_type, as: :radio_buttons, collection: [[t('.whitelist'), true], [t('.blacklist'), false]],
required: false, label: false, input_html: {class: 'radio'}, wrapper_html: {class: 'filter-type'}
          .clearfix
          = f.input :keywords, required: false, wrapper_html: {class: 'form-group keywords'},
input_html: {class: 'form-control', rows: 3, data: {role:'tagsinput'}}
          .clearfix
          = f.input :active, as: :boolean, input_html: {class: 'checkbox'}, wrapper_html: {class:
'filter-active'}
          = f.input :feed_id, as: :hidden
          .clearfix
      .row.top-margin
        .col-xs-2.col-xs-offset-8
          - if filter.persisted?
            = link_to filter_path(filter), method: :delete, remote: true, class: 'btn' do
              = t('.delete')
        .col-xs-2
          %button.btn.js-submit
            = t('.apply')
```

8.1.9.5.3. create.js.erb

```
$(document).ready(function(){
  var app = new App();
  $filter_form = $("<%= j render 'filter_form', filter: @filter %>");
  app.refresh_content($('.js-form-container'), $filter_form, function(){
    app.apply_tagsinput();
  });
  alert('filter created successfully');
});
```

8.1.9.5.4. index.html.haml

```
.secondary-view
  .panel
    .row
      .col-xs-8.col-xs-offset-2
        %h2
          = t('.filters')
        %p
          = t('.filter_out_certain_entries')
        .box
          .js-selection-form
            = render 'feed_selection', feeds: @feeds, feed: @feed
        .js-form-container
          = render 'filter_form', filter: @filter

= content_for :bottom_js do
```

```
  = javascript_include_tag "filters/filters"
```

8.1.9.5.5. refresh_all.js.erb

```
$(document).ready(function(){
  var app = new App();
  $filter_form = $("<%= j render 'filter_form', filter: @filter %>");
  app.refresh_content($('.js-form-container'), $filter_form, function(){
    $feed_selection = $("<%= j render 'feed_selection', feeds: @feeds, feed: @feed %>");
    app.refresh_content($('.js-selection-form'), $feed_selection, function(){
      app.apply_select2();
    });
  });
});
```

8.1.9.5.6. refresh_form.js.erb

```
$(document).ready(function(){
  var app = new App();
  $filter_form = $("<%= j render 'filter_form', filter: @filter %>");
  app.refresh_content($('.js-form-container'), $filter_form, function(){
    app.apply_tagsinput();
  });
});
```

8.1.9.5.7. update.js.erb

```
$(document).ready(function(){
  var app = new App();
  $filter_form = $("<%= j render 'filter_form', filter: @filter %>");
  app.refresh_content($('.js-form-container'), $filter_form, function(){
    app.apply_tagsinput();
  });
  alert('filter updated successfully');
});
```

8.1.9.6. followings/

8.1.9.6.1. _search_form.html.haml

```
= search_form_for q, url: followings_path, builder: SimpleForm::FormBuilder, remote: true, html:
{novalidate: true} do |f|
  .row
    .col-xs-12
      = f.input :name_cont, required: false, label: User.human_attribute_name(:name), placeholder:
t('.search_by_name'), input_html: {class: 'form-control'}, wrapper_html: {class: 'form-group'}
  .row
    .col-xs-12
```

```
    = f.input :email_cont, required: false, label: User.human_attribute_name(:email), placeholder:
t('.search_by_email'), input_html: {class: 'form-control'}
  .row.padding-bottom-1
    .col-xs-3.col-xs-offset-9
      %button.btn
        = t('.search')
```

8.1.9.6.2. _user_search_results.html.haml

```
.table-container
  - if users.empty?
    .error-message.text-center
      .glyphicon.glyphicon-exclamation-sign{aria: {hidden: 'true'}}
      = t('.no_users_found')
  - else
    %table.table
      - users.each do |user|
        %tr{id: "js-followings-user-#{user.id}"}
          %td
            = user.name
          %td
            = user.email
          %td.js-toggle-follow
            = toggle_follow_button current_user, user
```

8.1.9.6.3. create.js.erb

```
$(document).ready(function(){
  var app = new App();

  // toggle follow button on followings index
  $toggle_follow_button = $("<%= j toggle_follow_button @following.user, @following.followed
%>");
  app.refresh_content($('<%= "#js-followings-user-#{@following.followed.id}"
%>').find('.js-toggle-follow'), $toggle_follow_button);

  // add followed user to sidebar
  <% if Following.count == 1 %>
    // remove no_followings_found message
    $('.js-followed-users-container').empty();
  <% end %>
  $followed_user = $("<%= j render 'layouts/followed_user', following:
FollowingDecorator.decorate(@following) %>");
  $followed_user.css('display', 'none');
  $('.js-followed-users-container').append($followed_user);
  $followed_user.fadeIn('slow');
});
```

8.1.9.6.4. destroy.js.erb

```
$(document).ready(function(){
  var app = new App();

  // toggle follow button on followings index
  $toggle_follow_button = $("<%= j toggle_follow_button @following.user, @following.followed
%>");
  app.refresh_content($('<%= "#js-followings-user-#{@following.followed.id}"
%>').find('.js-toggle-follow'), $toggle_follow_button);

  // remove followed user from sidebar
  $('<%= "#js-following-#{@following.id}" %>').fadeOut('slow', function(){
    $(this).remove();
    <% if Following.count == 0 %>
      // refresh followed users container
      $followed_users = $("<%= j render 'layouts/followed_users', followings: [] %>");
      $('.js-followed-users-container').html($followed_users);
    <% end %>
  });
});
```

8.1.9.6.5. error.js.erb

```
alert('error');
```

8.1.9.6.6. index.html.haml

```
.secondary-view
  .panel
    .row
      .col-xs-8.col-xs-offset-2
        %h2
          = t('.follow_user')
        %p
          = t('.by_following_a_user')
        .box
          %h3
            = t('.search')
          .js-search-form-container
            = render 'search_form', q: @q
        .box
          %h3
            = t('.results')
          .js-results-container
            = render 'user_search_results', users: @users

= content_for :bottom_js do
  = javascript_include_tag "followings/followings"
```

8.1.9.6.7. index.js.erb

```
$(document).ready(function(){
  var app = new App();
  $user_search_results = $("<%= j render 'user_search_results', users: @users %>");
  app.refresh_content($('.js-results-container'), $user_search_results);
});
```

8.1.9.7. layouts/

8.1.9.7.1. _aside.html.haml

```
#aside
  .row
    .col-xs-12
      .box
        %ul.nav.nav-pills.nav-stacked
          %li
            = sidebar_link_to 'entries', controller_name
          %li
            = sidebar_link_to 'categories', controller_name
          %li
            = sidebar_link_to 'filters', controller_name
          %li
            = sidebar_link_to 'notifications', controller_name
  .row
    .col-xs-12
      .box.last
        .title.text-center
          %span
            = t('.following')
            = sidebar_link_to 'followings', controller_name do
              .btn.icon
                g
        .follow-panel.pre-scrollable.js-followed-users-container
          = render 'layouts/followed_users', followings: followings
```

8.1.9.7.2. _followed_user.html.haml

```
.row.user{id: "js-following-#{following.id}", class: "#{'selected' if @user && @user.id ==
following.object.followed.id}"}
  .col-xs-9
    = link_to entries_path(user_id: following.object.followed.id) do
      .name
        = following.followed
  .col-xs-2
    = link_to following_path(id: following.id), method: :delete, remote: true do
      .btn.sidebar-btn.icon
        e
```

8.1.9.7.3. _followed_users.html.haml

```
- if followings.empty?
  .error-message.text-center.top-spacing-2
    .glyphicon.glyphicon-exclamation-sign{aria: {hidden: 'true'}}
    = t('.no_followings_found')
- else
  - followings.each do |following|
    = render 'layouts/followed_user', following: following
```

8.1.9.7.4. _footer.html.haml

```
#footer
  .row
    .col-xs-3
      %ul.links
        %li
          %a{href: "#"}
            = t('.contact')
        %li
          %a{href: "#"}
            = t('.about')
        %li
          %a{href: "#"}
            = t('.guide')
    .col-xs-6
    .col-xs-2.col-xs-offset-1
      .content-decoration
  .footer-decoration
  .license
    = t('.license')
```

8.1.9.7.5. _header.html.haml

```
#header
  .row
    .col-xs-3
      .logo
        = image_tag "flavicon.svg", class: 'img-responsive'
    .col-xs-9
      .row
        .col-xs-9
          .row.brand
            .col-xs-6.col-xs-offset-4
              = image_tag "brand.svg"
              %p.text-center
                = t('.simple_feed_aggregator_with')
        .col-xs-3.dropdown
```

```
        .btn.pull-right.dropdown-toggle#user-dropdown{data: {toggle: "dropdown"}}
          %span
            = current_user.name
          %span.icon
            b
        %ul.dropdown-menu.dropdown-menu-right{aria: {labelledby: "user-dropdown"}}
          %li
            = link_to t('cancel_account'), user_registration_path, method: :delete, data: {confirm:
t('.are_you_sure')}
          %li
            = link_to t('.sign_out'), destroy_user_session_path, method: :delete
```

8.1.9.7.6. _notification.html.haml

```
.modal.fade{id: 'notification'}
  .modal-dialog
    .modal-content{class: "#{type}"}
      = message
```

8.1.9.7.7. application.html.haml

```
!!!
%html
  %head
    %title Socialfeed
    = stylesheet_link_tag    "fonticons"
    = stylesheet_link_tag    "application", :media => "all"
    = javascript_include_tag "application"
    = csrf_meta_tags
    %meta{name: "viewport", content: "width=device-width, initial-scale=1"}
  %body
    - if notice
      = render 'layouts/notification', message: notice, type: 'info'
    - if alert
      = render 'layouts/notification', message: alert, type: 'danger'

    .main-container
      .container-fluid
        = render 'layouts/header'
        .row
          .col-xs-3
            = render 'layouts/aside', followings: @followings
          .col-xs-9
            #content
              = yield
        = render 'layouts/footer'

    = yield :bottom_js
```

8.1.9.7.8. landing.html.haml

```
!!!
%html
  %head
    %title Socialfeed
    /[if lt IE 9]
    %script{src: "http://html5shim.googlecode/svn/trunk/html5.js", type: "text/javascript"}
    /[endif]
    = stylesheet_link_tag    "application", :media => "all"
    = javascript_include_tag "application"
    = csrf_meta_tags
    %meta{name: "viewport", content: "width=device-width, initial-scale=1"}
  %body
    - if notice
      = render 'layouts/notification', message: notice, type: 'info'
    - if alert
      = render 'layouts/notification', message: alert, type: 'danger'

    .landing
      .container-fluid
        .row.brand
          .col-xs-12
            = image_tag "brand.svg", class: 'img-responsive'
            %p.text-center.lead
              = t('.keep_up_with_your_interests')
        .row
          .col-sm-8.col-sm-offset-2.col-xs-10.col-xs-offset-1
            .box
              .row
                .col-xs-12
                  = render 'devise/registrations/new', resource: controller_name == 'registrations' ?
resource : User.new
              .row
                .col-xs-12
                  = render 'devise/sessions/new', resource: controller_name == 'sessions' ? resource :
User.new
            = image_tag "rssbg-lower.svg", class: 'img-responsive rss-lower'
          .col-lg-8.col-lg-offset-2.col-md-10.col-md-offset-1.col-xs-12.col-xs-offset-0
            .row
              .col-xs-10.col-xs-offset-1.col-sm-4.col-sm-offset-0
                .frame.center
                  = image_tag "box.svg", class: 'img-responsive'
                  .text-center
                    = t('.store_feeds_in_a_single_place')
              .col-xs-1.hidden-sm.hidden-md.hidden-lg
              .col-xs-10.col-xs-offset-1.col-sm-4.col-sm-offset-0
                .frame.center
                  = image_tag "newspaper.svg", class: 'img-responsive'
```

```
            .text-center
              = t('.stay_informed')
          .col-xs-1.hidden-sm.hidden-md.hidden-lg
          .col-xs-10.col-xs-offset-1.col-sm-4.col-sm-offset-0
            .frame.center
              = image_tag "crowd.svg", class: 'img-responsive'
              .text-center
                = t('.share_with_others')
          .col-xs-1.hidden-sm.hidden-md.hidden-lg
      .row
        = render 'layouts/footer'

  = yield :bottom_js
```

8.1.9.8. notification_mailer/

8.1.9.8.1. notify_entries.html.haml

```
%h1
  = t('.important_topic_mentioned_in_entry')

- @notifications.each do |notification|
  - unless notification[:entries].empty?
    %h3
      = t('.feed', feed: notification[:feed].name)
    = link_to t('.deactivate_notifications'), notifications_url(id: notification[:feed].id), target: '_blank'
    %br
    - notification[:entries].each do |entry|
      %h5
        = raw entry.title
      = entry.summary
      = entry.content
      %br
      = t('.published_on')
      = time_tag entry.published
      %br
      = link_to t('.view_website'), entry.url, target: '_blank'
    %hr
```

8.1.9.8.2. notify_entry.txt.erb

```
= t('.important_topic_mentioned_in_entry')

Title: <%= @entry.name %>
Summary: <%= @entry.summary %>
Content: <%= @entry.content %>
URL: <%= @entry.url %>
```

8.1.9.8.3. notify_following.html.haml

```
%h1
  = t('.another_user_is_following_you')

%p
  = t('.follower_name', name: @follower.name)
  %br
  = t('.follower_email', email: @follower.email)

%p
  = link_to t('.go_to_followings'), followings_url, target: '_blank'
```

8.1.9.9. notifications/

8.1.9.9.1. _feed_selection.html.haml

```
.row
  = form_tag notifications_path, method: :get, remote: true do |f|
    .col-xs-12
      = select_tag :id, options_from_collection_for_select(feeds, :id, :name, feed._?.id),
include_blank: true, class: 'form-control', data: {placeholder: t('placeholders.select_feed')}
```

8.1.9.9.2. _notification_form.html.haml

```
- if notification
  = simple_form_for notification, remote: true, html: {class: 'top-margin bottom-margin'} do |f|
    .row
      .col-xs-12
        = f.input :keywords, required: false, wrapper_html: {class: 'form-group keywords'}, input_html:
{class: 'form-control', rows: 3, data: {role:'tagsinput'}}
      .col-xs-12
        = f.input :active, as: :boolean, input_html: {class: 'checkbox'}, wrapper_html: {class:
'filter-active'}
        = f.input :feed_id, as: :hidden
    .row.top-margin
      .col-xs-2.col-xs-offset-8
        - if notification.persisted?
          = link_to notification_path(notification), method: :delete, remote: true, class: 'btn' do
            = t('.delete')
      .col-xs-2
        %button.btn.js-submit
          = t('.apply')
```

8.1.9.9.3. _user_notifications.html.haml

```
%h3
  = t('.general_notifications')
%p
  = t('.turn_general_user_level_notifications')
```

```
= simple_form_for user, url: url_for(action: 'update', controller: 'users/registrations'), method: :put,
remote: true do |f|
  .checkbox-group
    .row
      .col-xs-12
        = f.input :notify_new_follower, as: :boolean, wrapper_html: {class: 'form-group'}
    .row
      .col-xs-12
        = f.input :notify_site_updates, as: :boolean, wrapper_html: {class: 'form-group'}
    .row
      .col-xs-12
        = f.input :notify_important_topics, as: :boolean, wrapper_html: {class: 'form-group'}
  .row
    .col-xs-2.col-xs-offset-10
      %button.btn.js-submit
        = t('.apply')
```

8.1.9.9.4. create.js.erb

```
$(document).ready(function(){
  var app = new App();
  $notification_form = $("<%= j render 'notification_form', notification: @notification %>");
  app.refresh_content($('.js-form-container'), $notification_form, function(){
    app.apply_tagsinput();
  });
  alert('notification created successfully');
});
```

8.1.9.9.5. index.html.haml

```
.secondary-view
  .panel
    .row
      .col-xs-8.col-xs-offset-2
        %h2
          = t('.notifications')
        %p
          = t('.choose_keywords_for_each')
        .box
          .js-selection-form
            = render 'feed_selection', feeds: @feeds, feed: @feed
          .js-form-container
            = render 'notification_form', notification: @notification
        .box
          .js-checklist-container
            = render 'user_notifications', user: @user

= content_for :bottom_js do
  = javascript_include_tag "categories/categories"
```

8.1.9.9.6. refresh_all.js.erb

```
$(document).ready(function(){
  var app = new App();
  $notification_form = $("<%= j render 'notification_form', notification: @notification %>");
  app.refresh_content($('.js-form-container'), $notification_form, function(){
    $feed_selection = $("<%= j render 'feed_selection', feeds: @feeds, feed: @feed %>");
    app.refresh_content($('.js-selection-form'), $feed_selection, function(){
      app.apply_select2();
    });
  });
});
```

8.1.9.9.7. refresh_form.js.erb

```
$(document).ready(function(){
  var app = new App();
  $notification_form = $("<%= j render 'notification_form', notification: @notification %>");
  app.refresh_content($('.js-form-container'), $notification_form, function(){
    app.apply_tagsinput();
  });
});
```

8.1.9.9.8. refresh_user_notifications_form.js.erb

```
$(document).ready(function(){
  alert('General notifications successfully updated');
});
```

8.1.9.9.9. update.js.erb

```
$(document).ready(function(){
  var app = new App();
  $notification_form = $("<%= j render 'notification_form', notification: @notification %>");
  app.refresh_content($('.js-form-container'), $notification_form, function(){
    app.apply_tagsinput();
  });
  alert('notification updated successfully');
});
```

8.1.10. workers/

8.1.10.1. entry_worker.rb

```
class EntryWorker
  include Sidekiq::Worker
  include Sidetiq::Schedulable
```

```
  recurrence { daily.hour_of_day(0).minute_of_hour(1) }

  # daily removal of older entries
  def perform
    feed_ids = Feed.pluck('DISTINCT id')
    feed_ids.each do |feed_id|
                                cutoff = Entry.where(feed_id: feed_id).order('published_at
DESC').offset(50).first
                                Entry.where(feed_id: feed_id).deletable.where(['published_at <= ?',
cutoff.published_at]).delete_all
                        end
  rescue Exception => e
    logger.debug "EntryWorker:/n#{e.message}"
  end
end
```

8.1.10.2. feed_worker.rb

```
class FeedWorker
  include Sidekiq::Worker
  include Sidetiq::Schedulable

  recurrence { hourly.minute_of_hour(0, 15, 30, 45) }

  # update all feed's entries
  def perform
    User.find_each(batch_size: 10) do |user|
      notifications = []
      user.feeds.each do |feed|
        begin
          entries = feed.update_entries
          unless !user.notify_important_topics || entries.empty?
                notifications << {feed: feed, entries: entries}
          end
        rescue Exception => e
          logger.debug "FeedWorker:/n#{e.message}"
        end
      end
      unless notifications.empty?
        NotificationMailer.notify_entries(user, notifications).deliver
      end
    end
  end
end
```

8.2. config/

8.2.1. environments/

8.2.1.1. development.rb

```
Socialfeed::Application.configure do
  # Settings specified here will take precedence over those in config/application.rb

  # In the development environment your application's code is reloaded on
  # every request. This slows down response time but is perfect for development
  # since you don't have to restart the web server when you make code changes.
  config.cache_classes = false

  # Log error messages when you accidentally call methods on nil.
  config.whiny_nils = true

  # Set default URL
  config.action_mailer.default_url_options = { host: 'localhost', port: 3000 }

  # Show full error reports and disable caching
  config.consider_all_requests_local       = true
  config.action_controller.perform_caching = false

  # Don't care if the mailer can't send
  config.action_mailer.raise_delivery_errors = false

  # Open mails in browser tabs
  config.action_mailer.delivery_method = :letter_opener

  # Print deprecation notices to the Rails logger
  config.active_support.deprecation = :log

  # Only use best-standards-support built into browsers
  config.action_dispatch.best_standards_support = :builtin

  # Raise exception on mass assignment protection for Active Record models
  config.active_record.mass_assignment_sanitizer = :strict

  # Log the query plan for queries taking more than this (works
  # with SQLite, MySQL, and PostgreSQL)
  config.active_record.auto_explain_threshold_in_seconds = 0.5

  # Do not compress assets
  config.assets.compress = false

  # Expands the lines which load the assets
  config.assets.debug = true
end
```

8.2.1.2. production.rb

```
Socialfeed::Application.configure do
```

```
# Settings specified here will take precedence over those in config/application.rb

# Code is not reloaded between requests
config.cache_classes = true

# Full error reports are disabled and caching is turned on
config.consider_all_requests_local       = false
config.action_controller.perform_caching = true

# Disable Rails's static asset server (Apache or nginx will already do this)
config.serve_static_assets = false

# Compress JavaScripts and CSS
config.assets.compress = true

# Don't fallback to assets pipeline if a precompiled asset is missed
config.assets.compile = false

# Generate digests for assets URLs
config.assets.digest = true

# Devise configuration for Heroku
# config.assets.initialize_on_precompile = false

# Defaults to nil and saved in location specified by config.assets.prefix
# config.assets.manifest = YOUR_PATH

# Specifies the header that your server uses for sending files
# config.action_dispatch.x_sendfile_header = "X-Sendfile" # for apache
# config.action_dispatch.x_sendfile_header = 'X-Accel-Redirect' # for nginx

# Force all access to the app over SSL, use Strict-Transport-Security, and use secure cookies.
# config.force_ssl = true

# See everything in the log (default is :info)
# config.log_level = :debug

# Prepend all log lines with the following tags
# config.log_tags = [ :subdomain, :uuid ]

# Use a different logger for distributed setups
# config.logger = ActiveSupport::TaggedLogging.new(SyslogLogger.new)

# Use a different cache store in production
# config.cache_store = :mem_cache_store

# Enable serving of images, stylesheets, and JavaScripts from an asset server
# config.action_controller.asset_host = "http://assets.example.com"
```

```
  # Precompile additional assets (application.js, application.css, and all non-JS/CSS are already
added)
  # config.assets.precompile += %w( search.js )

  # Disable delivery errors, bad email addresses will be ignored
  # config.action_mailer.raise_delivery_errors = false

  # Enable threaded mode
  # config.threadsafe!

  # Enable locale fallbacks for I18n (makes lookups for any locale fall back to
  # the I18n.default_locale when a translation can not be found)
  config.i18n.fallbacks = true

  # Send deprecation notices to registered listeners
  config.active_support.deprecation = :notify

  # Log the query plan for queries taking more than this (works
  # with SQLite, MySQL, and PostgreSQL)
  # config.active_record.auto_explain_threshold_in_seconds = 0.5
end
```

8.2.1.3. test.rb

```
Socialfeed::Application.configure do
  # Settings specified here will take precedence over those in config/application.rb

  # The test environment is used exclusively to run your application's
  # test suite. You never need to work with it otherwise. Remember that
  # your test database is "scratch space" for the test suite and is wiped
  # and recreated between test runs. Don't rely on the data there!
  config.cache_classes = true

  # Set default URL
  config.action_mailer.default_url_options = { host: 'localhost', port: 3000 }

  # Configure static asset server for tests with Cache-Control for performance
  config.serve_static_assets = true
  config.static_cache_control = "public, max-age=3600"

  # Log error messages when you accidentally call methods on nil
  config.whiny_nils = true

  # Show full error reports and disable caching
  config.consider_all_requests_local       = true
  config.action_controller.perform_caching = false

  # Raise exceptions instead of rendering exception templates
  config.action_dispatch.show_exceptions = false
```

```
  # Disable request forgery protection in test environment
  config.action_controller.allow_forgery_protection    = false

  # Tell Action Mailer not to deliver emails to the real world.
  # The :test delivery method accumulates sent emails in the
  # ActionMailer::Base.deliveries array.
  config.action_mailer.delivery_method = :test

  # Raise exception on mass assignment protection for Active Record models
  config.active_record.mass_assignment_sanitizer = :strict

  # Print deprecation notices to the stderr
  config.active_support.deprecation = :stderr
end
```

8.2.2. initializers/

8.2.2.1. backtrace_silencers.rb

```
# Be sure to restart your server when you modify this file.

# You can add backtrace silencers for libraries that you're using but don't wish to see in your
backtraces.
# Rails.backtrace_cleaner.add_silencer { |line| line =~ /my_noisy_library/ }

# You can also remove all the silencers if you're trying to debug a problem that might stem from
framework code.
# Rails.backtrace_cleaner.remove_silencers!
```

8.2.2.2. devise.rb

```
# Use this hook to configure devise mailer, warden hooks and so forth.
# Many of these configuration options can be set straight in your model.
Devise.setup do |config|
  # The secret key used by Devise. Devise uses this key to generate
  # random tokens. Changing this key will render invalid all existing
  # confirmation, reset password and unlock tokens in the database.
  config.secret_key =
'ab75c46900fdcfac4ef3ef8fee6fbb42eab2a2a8e53a782bc23227613162f5caba3ffedcf3df93e8fc15
d2d2a72b60ad024750cb2bdc9991203c3c0fd85c09f3'

  # ==> Mailer Configuration
  # Configure the e-mail address which will be shown in Devise::Mailer,
  # note that it will be overwritten if you use your own mailer class
  # with default "from" parameter.
  config.mailer_sender = 'please-change-me-at-config-initializers-devise@example.com'

  # Configure the class responsible to send e-mails.
```

```
# config.mailer = 'Devise::Mailer'

# ==> ORM configuration
# Load and configure the ORM. Supports :active_record (default) and
# :mongoid (bson_ext recommended) by default. Other ORMs may be
# available as additional gems.
require 'devise/orm/active_record'

# ==> Configuration for any authentication mechanism
# Configure which keys are used when authenticating a user. The default is
# just :email. You can configure it to use [:username, :subdomain], so for
# authenticating a user, both parameters are required. Remember that those
# parameters are used only when authenticating and not when retrieving from
# session. If you need permissions, you should implement that in a before filter.
# You can also supply a hash where the value is a boolean determining whether
# or not authentication should be aborted when the value is not present.
# config.authentication_keys = [ :email ]

# Configure parameters from the request object used for authentication. Each entry
# given should be a request method and it will automatically be passed to the
# find_for_authentication method and considered in your model lookup. For instance,
# if you set :request_keys to [:subdomain], :subdomain will be used on authentication.
# The same considerations mentioned for authentication_keys also apply to request_keys.
# config.request_keys = []

# Configure which authentication keys should be case-insensitive.
# These keys will be downcased upon creating or modifying a user and when used
# to authenticate or find a user. Default is :email.
config.case_insensitive_keys = [ :email ]

# Configure which authentication keys should have whitespace stripped.
# These keys will have whitespace before and after removed upon creating or
# modifying a user and when used to authenticate or find a user. Default is :email.
config.strip_whitespace_keys = [ :email ]

# Tell if authentication through request.params is enabled. True by default.
# It can be set to an array that will enable params authentication only for the
# given strategies, for example, `config.params_authenticatable = [:database]` will
# enable it only for database (email + password) authentication.
# config.params_authenticatable = true

# Tell if authentication through HTTP Auth is enabled. False by default.
# It can be set to an array that will enable http authentication only for the
# given strategies, for example, `config.http_authenticatable = [:database]` will
# enable it only for database authentication. The supported strategies are:
# :database      = Support basic authentication with authentication key + password
# config.http_authenticatable = false

# If http headers should be returned for AJAX requests. True by default.
```

```
  # config.http_authenticatable_on_xhr = true

  # The realm used in Http Basic Authentication. 'Application' by default.
  # config.http_authentication_realm = 'Application'

  # It will change confirmation, password recovery and other workflows
  # to behave the same regardless if the e-mail provided was right or wrong.
  # Does not affect registerable.
  # config.paranoid = true

  # By default Devise will store the user in session. You can skip storage for
  # particular strategies by setting this option.
  # Notice that if you are skipping storage for all authentication paths, you
  # may want to disable generating routes to Devise's sessions controller by
  # passing skip: :sessions to `devise_for` in your config/routes.rb
  config.skip_session_storage = [:http_auth]

  # By default, Devise cleans up the CSRF token on authentication to
  # avoid CSRF token fixation attacks. This means that, when using AJAX
  # requests for sign in and sign up, you need to get a new CSRF token
  # from the server. You can disable this option at your own risk.
  # config.clean_up_csrf_token_on_authentication = true

  # ==> Configuration for :database_authenticatable
  # For bcrypt, this is the cost for hashing the password and defaults to 10. If
  # using other encryptors, it sets how many times you want the password re-encrypted.
  #
  # Limiting the stretches to just one in testing will increase the performance of
  # your test suite dramatically. However, it is STRONGLY RECOMMENDED to not use
  # a value less than 10 in other environments. Note that, for bcrypt (the default
  # encryptor), the cost increases exponentially with the number of stretches (e.g.
  # a value of 20 is already extremely slow: approx. 60 seconds for 1 calculation).
  config.stretches = Rails.env.test? ? 1 : 10

  # Setup a pepper to generate the encrypted password.
  # config.pepper =
'48fb86945d7f46b9b621b21cd1745083f693028908c911f838b7d5d4953d831229d5ea32631dc7d8
09a7844c64375587dbdd29c1fe488aca049eb9fd66800ccf'

  # ==> Configuration for :confirmable
  # A period that the user is allowed to access the website even without
  # confirming their account. For instance, if set to 2.days, the user will be
  # able to access the website for two days without confirming their account,
  # access will be blocked just in the third day. Default is 0.days, meaning
  # the user cannot access the website without confirming their account.
  # config.allow_unconfirmed_access_for = 2.days

  # A period that the user is allowed to confirm their account before their
  # token becomes invalid. For example, if set to 3.days, the user can confirm
```

```
# their account within 3 days after the mail was sent, but on the fourth day
# their account can't be confirmed with the token any more.
# Default is nil, meaning there is no restriction on how long a user can take
# before confirming their account.
# config.confirm_within = 3.days

# If true, requires any email changes to be confirmed (exactly the same way as
# initial account confirmation) to be applied. Requires additional unconfirmed_email
# db field (see migrations). Until confirmed, new email is stored in
# unconfirmed_email column, and copied to email column on successful confirmation.
config.reconfirmable = true

# Defines which key will be used when confirming an account
# config.confirmation_keys = [ :email ]

# ==> Configuration for :rememberable
# The time the user will be remembered without asking for credentials again.
# config.remember_for = 2.weeks

# Invalidates all the remember me tokens when the user signs out.
config.expire_all_remember_me_on_sign_out = true

# If true, extends the user's remember period when remembered via cookie.
# config.extend_remember_period = false

# Options to be passed to the created cookie. For instance, you can set
# secure: true in order to force SSL only cookies.
# config.rememberable_options = {}

# ==> Configuration for :validatable
# Range for password length.
config.password_length = 8..128

# Email regex used to validate email formats. It simply asserts that
# one (and only one) @ exists in the given string. This is mainly
# to give user feedback and not to assert the e-mail validity.
# config.email_regexp = /\A[^@]+@[^@]+\z/

# ==> Configuration for :timeoutable
# The time you want to timeout the user session without activity. After this
# time the user will be asked for credentials again. Default is 30 minutes.
# config.timeout_in = 30.minutes

# If true, expires auth token on session timeout.
# config.expire_auth_token_on_timeout = false

# ==> Configuration for :lockable
# Defines which strategy will be used to lock an account.
# :failed_attempts = Locks an account after a number of failed attempts to sign in.
```

```
# :none           = No lock strategy. You should handle locking by yourself.
# config.lock_strategy = :failed_attempts

# Defines which key will be used when locking and unlocking an account
# config.unlock_keys = [ :email ]

# Defines which strategy will be used to unlock an account.
# :email = Sends an unlock link to the user email
# :time  = Re-enables login after a certain amount of time (see :unlock_in below)
# :both  = Enables both strategies
# :none  = No unlock strategy. You should handle unlocking by yourself.
# config.unlock_strategy = :both

# Number of authentication tries before locking an account if lock_strategy
# is failed attempts.
# config.maximum_attempts = 20

# Time interval to unlock the account if :time is enabled as unlock_strategy.
# config.unlock_in = 1.hour

# Warn on the last attempt before the account is locked.
# config.last_attempt_warning = false

# ==> Configuration for :recoverable
#
# Defines which key will be used when recovering the password for an account
# config.reset_password_keys = [ :email ]

# Time interval you can reset your password with a reset password key.
# Don't put a too small interval or your users won't have the time to
# change their passwords.
config.reset_password_within = 6.hours

# ==> Configuration for :encryptable
# Allow you to use another encryption algorithm besides bcrypt (default). You can use
# :sha1, :sha512 or encryptors from others authentication tools as :clearance_sha1,
# :authlogic_sha512 (then you should set stretches above to 20 for default behavior)
# and :restful_authentication_sha1 (then you should set stretches to 10, and copy
# REST_AUTH_SITE_KEY to pepper).
#
# Require the `devise-encryptable` gem when using anything other than bcrypt
# config.encryptor = :sha512

# ==> Scopes configuration
# Turn scoped views on. Before rendering "sessions/new", it will first check for
# "users/sessions/new". It's turned off by default because it's slower if you
# are using only default views.
# config.scoped_views = false
```

```
# Configure the default scope given to Warden. By default it's the first
# devise role declared in your routes (usually :user).
# config.default_scope = :user

# Set this configuration to false if you want /users/sign_out to sign out
# only the current scope. By default, Devise signs out all scopes.
# config.sign_out_all_scopes = true

# ==> Navigation configuration
# Lists the formats that should be treated as navigational. Formats like
# :html, should redirect to the sign in page when the user does not have
# access, but formats like :xml or :json, should return 401.
#
# If you have any extra navigational formats, like :iphone or :mobile, you
# should add them to the navigational formats lists.
#
# The "*/*" below is required to match Internet Explorer requests.
# config.navigational_formats = ['*/*', :html]

# The default HTTP method used to sign out a resource. Default is :delete.
config.sign_out_via = :delete

# ==> OmniAuth
# Add a new OmniAuth provider. Check the wiki for more information on setting
# up on your models and hooks.
# config.omniauth :github, 'APP_ID', 'APP_SECRET', scope: 'user,public_repo'

# ==> Warden configuration
# If you want to use other strategies, that are not supported by Devise, or
# change the failure app, you can configure them inside the config.warden block.
#
# config.warden do |manager|
#   manager.intercept_401 = false
#   manager.default_strategies(scope: :user).unshift :some_external_strategy
# end

# ==> Mountable engine configurations
# When using Devise inside an engine, let's call it `MyEngine`, and this engine
# is mountable, there are some extra configurations to be taken into account.
# The following options are available, assuming the engine is mounted as:
#
#     mount MyEngine, at: '/my_engine'
#
# The router that invoked `devise_for`, in the example above, would be:
# config.router_name = :my_engine
#
# When using omniauth, Devise cannot automatically set Omniauth path,
# so you need to do it manually. For the users scope, it would be:
# config.omniauth_path_prefix = '/my_engine/users/auth'
```

```
end
```

8.2.2.3. inflections.rb

```
# Be sure to restart your server when you modify this file.

# Add new inflection rules using the following format
# (all these examples are active by default):
# ActiveSupport::Inflector.inflections do |inflect|
#   inflect.plural /^(ox)$/i, '\1en'
#   inflect.singular /^(ox)en/i, '\1'
#   inflect.irregular 'person', 'people'
#   inflect.uncountable %w( fish sheep )
# end
#
# These inflection rules are supported but not enabled by default:
# ActiveSupport::Inflector.inflections do |inflect|
#   inflect.acronym 'RESTful'
# end
```

8.2.2.4. mime_types.rb

```
# Be sure to restart your server when you modify this file.

# Add new mime types for use in respond_to blocks:
# Mime::Type.register "text/richtext", :rtf
# Mime::Type.register_alias "text/html", :iphone
```

8.2.2.5. secret_token.rb

```
# Be sure to restart your server when you modify this file.

# Your secret key for verifying the integrity of signed cookies.
# If you change this key, all old signed cookies will become invalid!
# Make sure the secret is at least 30 characters and all random,
# no regular words or you'll be exposed to dictionary attacks.
Socialfeed::Application.config.secret_token =
'3ce5dd1fbe14d8331fd3d15f57bf836ae75d9855fbd3173901544c9977a36415dd0db7280bd75f3cc
3293fab5a5527b5eee5f7200de2a69221d9bf855286b3f3'
```

8.2.2.6. session_store.rb

```
# Be sure to restart your server when you modify this file.

Socialfeed::Application.config.session_store :cookie_store, key: '_socialfeed_session'

# Use the database for sessions instead of the cookie-based default,
# which shouldn't be used to store highly confidential information
# (create the session table with "rails generate session_migration")
```

```
# Socialfeed::Application.config.session_store :active_record_store
```

8.2.2.7. wrap_parameters.rb

```
# Be sure to restart your server when you modify this file.
#
# This file contains settings for ActionController::ParamsWrapper which
# is enabled by default.

# Enable parameter wrapping for JSON. You can disable this by setting :format to an empty array.
ActiveSupport.on_load(:action_controller) do
  wrap_parameters format: [:json]
end

# Disable root element in JSON by default.
ActiveSupport.on_load(:active_record) do
  self.include_root_in_json = false
end
```

8.2.3. locales/

8.2.3.1. devise.en.yml

```
---
en:
  errors:
    messages:
      already_confirmed: was already confirmed, please try signing in
      confirmation_period_expired: needs to be confirmed within %{period}, please request a new
one
      expired: has expired, please request a new one
      not_found: not found
      not_locked: was not locked
      not_saved:
        one: '1 error prohibited this %{resource} from being saved:'
        other: "%{count} errors prohibited this %{resource} from being saved:"
  notifications:
    user_notifications:
      turn_general_user_level_notifications: Turn general user level notifications on or off. If you turn
off the important topics notification you won't receive any notifications about entries, even if they
are set.
  devise:
    confirmations:
      confirmed: Your email address has been successfully confirmed.
      send_instructions: You will receive an email with instructions for how to confirm your email
address in a few minutes.
      send_paranoid_instructions: If your email address exists in our database, you will receive an
email with instructions for how to confirm your email address in a few minutes.
    failure:
```

```
    already_authenticated: You are already signed in.
    inactive: Your account is not activated yet.
    invalid: Invalid email or password.
    locked: Your account is locked.
    last_attempt: You have one more attempt before your account is locked.
    not_found_in_database: Invalid email address or password.
    timeout: Your session expired. Please sign in again to continue.
    unauthenticated: You need to sign in or sign up before continuing.
    unconfirmed: You have to confirm your email address before continuing.
    user:
      not_found_in_database: User could not be found.
  mailer:
    confirmation_instructions:
      subject: Confirmation instructions
    reset_password_instructions:
      subject: Reset password instructions
    unlock_instructions:
      subject: Unlock instructions
  omniauth_callbacks:
    failure: Could not authenticate you from %{kind} because "%{reason}".
    success: Successfully authenticated from %{kind} account.
  passwords:
    no_token: You can't access this page without coming from a password reset email. If you do
come from a password reset email, please make sure you used the full URL provided.
    send_instructions: You will receive an email with instructions on how to reset your password in
a few minutes.
    send_paranoid_instructions: If your email address exists in our database, you will receive a
password recovery link at your email address in a few minutes.
    updated: Your password has been changed successfully. You are now signed in.
    updated_not_active: Your password has been changed successfully.
  registrations:
    destroyed: Bye! Your account has been successfully cancelled. We hope to see you again
soon.
    signed_up: Welcome! You have signed up successfully.
    signed_up_but_inactive: You have signed up successfully. However, we could not sign you in
because your account is not yet activated.
    signed_up_but_locked: You have signed up successfully. However, we could not sign you in
because your account is locked.
    signed_up_but_unconfirmed: A message with a confirmation link has been sent to your email
address. Please follow the link to activate your account.
    update_needs_confirmation: You updated your account successfully, but we need to verify
your new email address. Please check your email and follow the confirm link to confirm your new
email address.
    updated: Your account has been updated successfully.
    new:
      back: Back
      sign_up: Sign up
      pick_a_username: Pick a username
      reenter_password: Reenter password
```

```
  sessions:
    signed_in: Signed in successfully.
    signed_out: Signed out successfully.
    already_signed_out: Signed out successfully.
    new:
      already_got_an_account: Already got an account?
      sign_in: Sign in
      remember_me: Remember me
  shared:
    links:
      forgot_your_password: Forgot your password
  unlocks:
    send_instructions: You will receive an email with instructions for how to unlock your account in a few minutes.
    send_paranoid_instructions: If your account exists, you will receive an email with instructions for how to unlock it in a few minutes.
    unlocked: Your account has been unlocked successfully. Please sign in to continue.
```

8.2.3.2. devise.es.yml

```
---
es:
  errors:
    messages:
      already_confirmed: ya ha sido confirmado, por favor intente iniciar sesión.
      confirmation_period_expired: necesita ser confirmado en %{period}, por favor vuelva a solicitarla.
      expired: ha expirado, por favor vuelva a solicitarla.
      not_found: no se ha encontrado.
      not_locked: no estaba bloqueado.
      not_saved:
        one: '1 error impidió que este %{resource} fuera guardado:'
        other: "%{count} errores impidieron que este %{resource} fuera guardado:"
  notifications:
    user_notifications:
      turn_general_user_level_notifications: Active o desactive notificaciones generales a nivel de usuario. Si desactiva las notificaciones de temas importantes no recibirá notificaciones sobre entradas, aún cuando estas estén establecidas.
  devise:
    confirmations:
      confirmed: Su cuenta ha sido confirmada.
      send_instructions: Recibirá un correo electrónico en unos minutos con instrucciones sobre cómo restablecer su contraseña.
      send_paranoid_instructions: Si su correo electrónico existe en nuestra base de datos recibirá un correo electrónico en unos minutos con instrucciones sobre cómo restablecer su contraseña.
    failure:
      already_authenticated: Ya ha iniciado sesión.
      inactive: Su cuenta aún no ha sido activada.
      invalid: Dirección de email o contraseña inválidas.
```

```
    locked: Su cuenta ha sido bloqueada.
    last_attempt: Tiene un último intento antes de que su cuenta sea bloqueada.
    not_found_in_database: Correo o contraseña inválidos.
    timeout: Su sesión ha expirado, por favor inicie sesión nuevamente para continuar.
    unauthenticated: Necesita iniciar sesión o registrarse para continuar.
    unconfirmed: Debe confirmar su cuenta para continuar.
    user:
      not_found_in_database: No se pudo encontrar el usuario.
  mailer:
    confirmation_instructions:
      subject: Instrucciones de confirmación
    reset_password_instructions:
      subject: Instrucciones para restablecer su contraseña
    unlock_instructions:
      subject: Instrucciones de desbloqueo
  omniauth_callbacks:
    failure: No se le ha podido autorizar de %{kind} debido a "%{reason}".
    success: Identificado correctamente de %{kind}.
  passwords:
    no_token: No puede acceder a esta página sino es a través de un enlace para restablecer la
contraseña. Si ha accedido desde el enlace para restablecer la contraseña, asegúrese de que la
URL esté completa.
    send_instructions: Recibirá un correo electrónico con instrucciones sobre cómo restablecer su
contraseña en unos minutos.
    send_paranoid_instructions: Si su correo electrónico existe en nuestra base de datos, recibirá
un enlace para restablecer la contraseña en unos minutos.
    updated: Su contraseña ha cambiado correctamente. Ha sido identificado correctamente.
    updated_not_active: Su contraseña se ha cambiado correctamente.
  registrations:
    destroyed: "¡Adiós! Su cuenta ha sido cancelada. Esperamos volver a verlo pronto."
    signed_up: "¡Bienvenido! Usted ha sido identificado."
    signed_up_but_inactive: Se ha registrado correctamente, pero no ha podido iniciar sesión
porque su cuenta no ha sido activada.
    signed_up_but_locked: Se ha registrado correctamente, pero no ha podido iniciar sesión
porque su cuenta está bloqueada.
    signed_up_but_unconfirmed: Se le ha enviado un mensaje con un enlace de confirmación.
Por favor visite el enlace para activar su cuenta.
    update_needs_confirmation: Ha actualizado su cuenta correctamente, sin embargo
necesitamos verificar su nueva cuenta de correo. Por favor revise su correo electrónico y visite el
enlace para finalizar la confirmación de su nueva dirección de correo electrónico.
    updated: Ha actualizado tu cuenta correctamente.
    new:
      back: Atrás
      sign_up: Registrarse
      pick_a_username: Elija un nombre de usuario
      reenter_password: Reingrese la contraseña
  sessions:
    signed_in: Ha iniciado sesión correctamente.
    signed_out: Ha cerrado la sesión correctamente.
```

```
      already_signed_out: Ha cerrado la sesión correctamente.
      new:
        already_got_an_account: Ya tiene una cuenta?
        sign_in: Sign in
        remember_me: Recordarme
    shared:
      links:
        forgot_your_password: Olvidó su contraseña?
    unlocks:
      send_instructions: Recibirá un correo electrónico en unos minutos con instrucciones sobre
cómo desbloquear su cuenta.
      send_paranoid_instructions: Si su cuenta existe, recibirá un correo electrónico en unos
minutos con instrucciones sobre cómo desbloquear su cuenta.
      unlocked: Su cuenta ha sido desbloqueada. Por favor inicie sesión para continuar.
```

8.2.3.3. en.bootstrap.yml

```
# Sample localization file for English. Add more files in this directory for other locales.
# See https://github.com/svenfuchs/rails-i18n/tree/master/rails%2Flocale for starting points.

en:
  helpers:
    actions: "Actions"
    links:
      back: "Back"
      cancel: "Cancel"
      confirm: "Are you sure?"
      destroy: "Delete"
      new: "New"
      edit: "Edit"
    titles:
      edit: "Edit %{model}"
      save: "Save %{model}"
      new: "New %{model}"
      delete: "Delete %{model}"
```

8.2.3.4. en.yml

```
---
en:
  errors:
    messages:
      accepted: must be accepted
      blank: can't be blank
      present: must be blank
      confirmation: doesn't match %{attribute}
      empty: can't be empty
      equal_to: must be equal to %{count}
      even: must be even
```

```
    exclusion: is reserved
    greater_than: must be greater than %{count}
    greater_than_or_equal_to: must be greater than or equal to %{count}
    inclusion: is not included in the list
    invalid: is invalid
    less_than: must be less than %{count}
    less_than_or_equal_to: must be less than or equal to %{count}
    not_a_number: is not a number
    not_an_integer: must be an integer
    odd: must be odd
    record_invalid: 'Validation failed: %{errors}'
    restrict_dependent_destroy:
      one: Cannot delete record because a dependent %{record} exists
      many: Cannot delete record because dependent %{record} exist
    taken: has already been taken
    too_long:
      one: is too long (maximum is 1 character)
      other: is too long (maximum is %{count} characters)
    too_short:
      one: is too short (minimum is 1 character)
      other: is too short (minimum is %{count} characters)
    wrong_length:
      one: is the wrong length (should be 1 character)
      other: is the wrong length (should be %{count} characters)
    other_than: must be other than %{count}
    invalid_feed_url: URL inválida
    could_not_process: 'Could not process your petition, please try again'
  format: "%{attribute} %{message}"
  template:
    body: 'There were problems with the following fields:'
    header:
      one: 1 error prohibited this %{model} from being saved
      other: "%{count} errors prohibited this %{model} from being saved"
  can_not_be_destroyed: Can not be destroyed
  can_not_be_edited: Can not be edited
notifications:
  user_notifications:
    apply: Apply
    general_notifications: General notifications
  notification_form:
    delete: Delete
    apply: Apply
  index:
    notifications: Notifications
    choose_keywords_for_each: Choose keywords for each of your feeds. We willsend you an
email when a matching feed-entry shows up.
date:
  abbr_day_names:
  - Sun
```

- Mon
- Tue
- Wed
- Thu
- Fri
- Sat
abbr_month_names:
-
- Jan
- Feb
- Mar
- Apr
- May
- Jun
- Jul
- Aug
- Sep
- Oct
- Nov
- Dec
day_names:
- Sunday
- Monday
- Tuesday
- Wednesday
- Thursday
- Friday
- Saturday
formats:
  default: "%Y-%m-%d"
  long: "%B %d, %Y"
  short: "%b %d"
month_names:
-
- January
- February
- March
- April
- May
- June
- July
- August
- September
- October
- November
- December
order:
- :year
- :month

```
  - :day
datetime:
  distance_in_words:
    about_x_hours:
      one: about an hour ago
      other: about %{count} hours ago
    about_x_months:
      one: about a month ago
      other: about %{count} months ago
    about_x_years:
      one: about a year ago
      other: about %{count} years ago
    almost_x_years:
      one: almost a year ago
      other: almost %{count} years ago
    half_a_minute: half a minute ago
    less_than_x_minutes:
      one: less than a minute ago
      other: less than %{count} minutes ago
    less_than_x_seconds:
      one: less than a second ago
      other: less than %{count} seconds ago
    over_x_years:
      one: over a year ago
      other: over %{count} years ago
    x_days:
      one: a day ago
      other: "%{count} days ago"
    x_minutes:
      one: a minute ago
      other: "%{count} minutes ago"
    x_months:
      one: a month ago
      other: "%{count} months ago"
    x_seconds:
      one: a second ago
      other: "%{count} seconds ago"
  prompts:
    day: Day
    hour: Hour
    minute: Minute
    month: Month
    second: Seconds
    year: Year
helpers:
  select:
    prompt: Please select
  submit:
    create: Create %{model}
```

```
    submit: Save %{model}
    update: Update %{model}
  number:
    currency:
      format:
        delimiter: ","
        format: "%u%n"
        precision: 2
        separator: "."
        significant: false
        strip_insignificant_zeros: false
        unit: "$"
    format:
      delimiter: ","
      precision: 3
      separator: "."
      significant: false
      strip_insignificant_zeros: false
    human:
      decimal_units:
        format: "%n %u"
        units:
          billion: Billion
          million: Million
          quadrillion: Quadrillion
          thousand: Thousand
          trillion: Trillion
          unit: ''
      format:
        delimiter: ''
        precision: 3
        significant: true
        strip_insignificant_zeros: true
      storage_units:
        format: "%n %u"
        units:
          byte:
            one: Byte
            other: Bytes
          gb: GB
          kb: KB
          mb: MB
          tb: TB
    percentage:
      format:
        delimiter: ''
        format: "%n%"
    precision:
      format:
```

```
      delimiter: ''
  support:
    array:
      last_word_connector: ", and "
      two_words_connector: " and "
      words_connector: ", "
  time:
    am: am
    formats:
      default: "%a, %d %b %Y %H:%M:%S %z"
      long: "%B %d, %Y %H:%M"
      short: "%d %b %H:%M"
    pm: pm
  hello: Hello world
  placeholders:
    untitled: Untitled
    undefined: Undefined
    read: Read
    unread: Unread
    all: All
    select_feed: Select feed
  layouts:
    aside:
      all: All
      saved: Saved
      categories: Categories
      filters: Filters
      notifications: Notifications
      following: Following
      entries: Home
      no_followings_found: No users are being followed
    header:
      sign_out: Sign out
      simple_feed_aggregator_with: Simple feed-aggregator with social functionality
      preferences: Preferences
      are_you_sure: Are you sure you wish to cancel your account?
    footer:
      contact: Contact
      about: About
      guide: Guide
      license: Copyright © 2015 MIT
    followed_users:
      no_followings_found: No followings found
    landing:
      keep_up_with_your_interests: Keep up with your interests and those of whom you care about
      store_feeds_in_a_single_place: Store feeds in a single place
      stay_informed: Stay informed
      share_with_others: Share with others
  feeds:
```

```
    feed_form:
      add_feed: Add feed
      cancel: Cancel
      add: Add
      adding: Adding...
    form_fields:
      paste_url_here: Paste the URL here
      give_custom_title: Give this feed a custom title
      add_description: Add a description
  entries:
    entry:
      view_website: View website
    entries_search_form:
      all: All
      read: Read
      unread: Unread
      favorite: Favorite
      not_favorite: Not favorite
    category_choice_form:
      all: All
    index:
      no_entries_found: No entries found
  categories:
    category_form:
      enter_category_name: Enter category name
      enter_description: Enter description
      enter_tags: Enter tags
      new: New
      edit: Edit
    index:
      categories: Categories
      create_a_new_category: Create or edit categories, view existing ones with their assigned
feeds or delete them.
      new: New
      list: List
      description: Description
      tags: Tags
    category:
      are_you_sure: This category has feeds associated with it, do you wish to delete it anyways?
  filters:
    index:
      filters: Filters
      filter_out_certain_entries: Filter out certain entries based on their content. Select a feed and
one of the following filtering options.
    filter_form:
      enter_keywords: Enter keywords
      whitelist: Whitelist
      blacklist: Blacklist
      apply: Apply
```

```
    delete: Delete
notification_mailer:
  notify_entries:
    important_topic_mentioned_in_entry: Important topic mentioned in entry
    deactivate_notifications: Deactivate this notification
    feed: 'Feed: %{feed}'
    view_website: View website
    published_on: Published on
    important_topic_notification: Important Topic Notification
    following_notification: Following Notification
  notify_entry:
    important_topic_mentioned_in_entry: Important topic mentioned in entry
  notify_following:
    another_user_is_following_you: Another user is following you
    follower_name: Follower name
    follower_email: Follower email
    go_to_followings: Go to followings
activerecord:
  models:
    category:
      one: Category
      other: Categories
    entry:
      one: Entry
      other: Entries
    feed:
      one: Feed
      other: Feeds
    filter:
      one: Filter
      other: Filters
    notification:
      one: Notification
      other: Notifications
    user:
      one: User
      other: Users
  attributes:
    category:
      name: Name
      tags: Tags
      description: Description
    entry:
      name: Name
      summary: Summary
      content: Content
      url: URL
      published_at: Published at
      read: Leída
```

```
      unread: No leída
    feed:
      name: Name
      url: URL
      description: Description
    filter:
      active: Active
      keywords: Keywords
      list_type: List Type
    notification:
      keywords: Keywords
      active: Active
    user:
      email: Email
      name: Name
      notify_new_follower: Notify new follower
      notify_site_updates: Notify site updates
      notify_important_topics: Notify important topics
      password: Contraseña
  followings:
    search_form:
      search_by_name: Search by name
      search_by_email: Search by email
      search: Search
    index:
      follow_user: Follow user
      by_following_a_user: By following a user
      search: Search
      results: Results
    user_search_results:
      no_users_found: No users found
  cancel_account: Cancel Account
```

8.2.3.5. es.yml

```
---
es:
  errors:
    messages:
      accepted: debe ser aceptado
      blank: no puede estar en blanco
      confirmation: no coincide
      empty: no puede estar vacío
      equal_to: debe ser igual a %{count}
      even: debe ser par
      exclusion: está reservado
      greater_than: debe ser mayor que %{count}
      greater_than_or_equal_to: debe ser mayor que o igual a %{count}
      inclusion: no está incluido en la lista
```

```
    invalid: no es válido
    invalid_feed_url: URL inválida
    less_than: debe ser menor que %{count}
    less_than_or_equal_to: debe ser menor que o igual a %{count}
    not_a_number: no es un número
    not_an_integer: debe ser un entero
    odd: debe ser impar
    record_invalid: 'La validación falló: %{errors}'
    taken: ya está en uso
    too_long: es demasiado largo (%{count} caracteres máximo)
    too_short: es demasiado corto (%{count} caracteres mínimo)
    wrong_length: no tiene la longitud correcta (%{count} caracteres exactos)
    present: must be blank
    restrict_dependent_destroy:
      one: Cannot delete record because a dependent %{record} exists
      many: Cannot delete record because dependent %{record} exist
    other_than: must be other than %{count}
    could_not_process: 'No se pudo procesar su petición, por favor intente nuevamente'
  format: "%{attribute} %{message}"
  template:
    body: 'Se encontraron problemas con los siguientes campos:'
    header:
      one: No se pudo guardar este/a %{model} porque se encontró 1 error
      other: No se pudo guardar este/a %{model} porque se encontraron %{count} errores
  can_not_be_destroyed: No puede ser borrado
  can_not_be_edited: No puede ser editado
 notifications:
  user_notifications:
    apply: Aplicar
    general_notifications: Notificaciones Generales
  notification_form:
    delete: Borrar
    apply: Aplicar
  index:
    notifications: Notificaciones
    choose_keywords_for_each: Elija palabras clave para cada uno de sus feeds. Le enviaremos
un email cuando aparezca una entrada que contenga alguna de las palabras clave.
 activerecord:
  models:
    filter:
      one: Filtro
      other: Filtros
    notification:
      one: Notificación
      other: Notificaciones
    user:
      one: Usuario
      other: Usuarios
    category:
```

```
      one: Categoría
      other: Categorías
    entry:
      one: Entrada
      other: Entradas
    feed:
      one: Feed
      other: Feeds
  attributes:
    filter:
      keywords: Palabras clave
      list_type: Tipo de lista
      active: Activo
    notification:
      keywords: Palabras clave
      active: Activo
    user:
      email: Email
      name: Nombre
      password: Contraseña
      notify_new_follower: Notificar nuevo seguidor
      notify_site_updates: Notificar cambios del sitio
      notify_important_topics: Notificar temas importantes
    category:
      name: Nombre
      tags: Etiquetas
      description: Descripción
    entry:
      name: Nombre
      summary: Resumen
      content: Contenido
      url: URL
      published_at: Publicado el
      read: Leída
      unread: No leída
    feed:
      name: Nombre
      url: URL
      description: Descripción
date:
  abbr_day_names:
  - dom
  - lun
  - mar
  - mié
  - jue
  - vie
  - sáb
  abbr_month_names:
```

```
  -
  - ene
  - feb
  - mar
  - abr
  - may
  - jun
  - jul
  - ago
  - sep
  - oct
  - nov
  - dic
  day_names:
  - domingo
  - lunes
  - martes
  - miércoles
  - jueves
  - viernes
  - sábado
  formats:
    default: "%d/%m/%Y"
    long: "%d de %B de %Y"
    short: "%d de %b"
  month_names:
  -
  - enero
  - febrero
  - marzo
  - abril
  - mayo
  - junio
  - julio
  - agosto
  - septiembre
  - octubre
  - noviembre
  - diciembre
  order:
  - :day
  - :month
  - :year
datetime:
  distance_in_words:
    about_x_hours:
      one: alrededor de 1 hora
      other: alrededor de %{count} horas
    about_x_months:
```

```
      one: alrededor de 1 mes
      other: alrededor de %{count} meses
    about_x_years:
      one: alrededor de 1 año
      other: alrededor de %{count} años
    almost_x_years:
      one: casi 1 año
      other: casi %{count} años
    half_a_minute: medio minuto
    less_than_x_minutes:
      one: menos de 1 minuto
      other: menos de %{count} minutos
    less_than_x_seconds:
      one: menos de 1 segundo
      other: menos de %{count} segundos
    over_x_years:
      one: más de 1 año
      other: más de %{count} años
    x_days:
      one: 1 día
      other: "%{count} días"
    x_minutes:
      one: 1 minuto
      other: "%{count} minutos"
    x_months:
      one: 1 mes
      other: "%{count} meses"
    x_seconds:
      one: 1 segundo
      other: "%{count} segundos"
  prompts:
    day: Día
    hour: Hora
    minute: Minutos
    month: Mes
    second: Segundos
    year: Año
helpers:
  select:
    prompt: Por favor seleccione
  submit:
    create: Crear %{model}
    submit: Guardar %{model}
    update: Actualizar %{model}
number:
  currency:
    format:
      delimiter: "."
      format: "%n %u"
```

```
      precision: 2
      separator: ","
      significant: false
      strip_insignificant_zeros: false
      unit: "€"
  format:
    delimiter: "."
    precision: 3
    separator: ","
    significant: false
    strip_insignificant_zeros: false
  human:
    decimal_units:
      format: "%n %u"
      units:
        billion: mil millones
        million: millón
        quadrillion: mil billones
        thousand: mil
        trillion: billón
        unit: ''
    format:
      delimiter: ''
      precision: 1
      significant: true
      strip_insignificant_zeros: true
    storage_units:
      format: "%n %u"
      units:
        byte:
          one: Byte
          other: Bytes
        gb: GB
        kb: KB
        mb: MB
        tb: TB
  percentage:
    format:
      delimiter: ''
      format: "%n%"
  precision:
    format:
      delimiter: ''
support:
  array:
    last_word_connector: ", y "
    two_words_connector: " y "
    words_connector: ", "
time:
```

```
  am: am
  formats:
    default: "%A, %d de %B de %Y %H:%M:%S %z"
    long: "%d de %B de %Y %H:%M"
    short: "%d de %b %H:%M"
  pm: pm
hello: Hello world
placeholders:
  untitled: Sin título
  undefined: Indefinido
  read: Leído
  unread: No Leído
  all: Todos
  select_feed: Seleccione un feed
layouts:
  aside:
    all: Todos
    saved: Guardado
    categories: Categorías
    filters: Filtros
    notifications: Notificaciones
    following: Siguiendo a
    entries: Inicio
    no_followings_found: No se encontraron seguimientos
  header:
    sign_out: Desconectar
    simple_feed_aggregator_with: Agregador de feeds simple con funcionalidades sociales
    preferences: Preferencias
    are_you_sure: Está seguro que desea cancelar su cuenta?
  footer:
    contact: Contacto
    about: Información
    guide: Guía
    license: Copyright © 2015 MIT
  followed_users:
    no_followings_found: No se está siguiendo a ningún usuario
  landing:
    keep_up_with_your_interests: Manténgase al tanto con sus intereses y aquellos de los demás
    store_feeds_in_a_single_place: Almacene feeds en un único lugar
    stay_informed: Manténgase informado
    share_with_others: Comparta con otros
feeds:
  feed_form:
    add_feed: Agregar feed
    cancel: Cancelar
    add: Agregar
    adding: Agregando...
  form_fields:
    paste_url_here: Pegar la URL aquí
```

```
    give_custom_title: Dar un título personalizado a este feed
    add_description: Agregar una descripción
  entries:
    entry:
      view_website: Ver en sitio web
    entries_search_form:
      all: Todas
      read: Leídas
      unread: No Leídas
      favorite: Favoritas
      not_favorite: No favoritas
    category_choice_form:
      all: Todas
    index:
      no_entries_found: No se encontraron entradas
  categories:
    category_form:
      enter_category_name: Ingrese el nombre de la categoría
      enter_description: Ingrese una descripción
      enter_tags: Ingrese etiquetas
      new: Nueva
      edit: Editar
    index:
      categories: Categorías
      create_a_new_category: Crear o editar categorías, ver las existentes con sus feeds asignados
or eliminarlas.
      new: Nueva
      list: Lista
      description: Descripción
      tags: Etiquetas
    category:
      are_you_sure: Esta categoría tiene feeds asociados, seguro que desea borrarla?
  filters:
    index:
      filters: Filtros
      filter_out_certain_entries: Filtrar ciertas entradas en base al contenido. Seleccione un feed y
una de las siguientes opciones de filtrado.
    filter_form:
      enter_keywords: Ingrese palabras clave
      whitelist: Lista Blanca
      blacklist: Lista Negra
      apply: Aplicar
      delete: Borrar
  notification_mailer:
    notify_entries:
      important_topic_mentioned_in_entry: Temas importantes mencionados en las entradas
      deactivate_notifications: Desactivar esta notificación
      feed: 'Feed: %{feed}'
      view_website: Ver en sitio web
```

```
      published_on: Publicado el
      important_topic_notification: Notificación de Tema Importante
      following_notification: Notificación de Seguimiento
    notify_entry:
      important_topic_mentioned_in_entry: Temas importantes mencionados en las entradas
    notify_following:
      another_user_is_following_you: Otro usuario lo está siguiendo
      follower_name: El nombre del seguidor es %{name}
      follower_email: La dirección de email del seguidor es %{email}
      go_to_followings: Ir a seguir usuario
  followings:
    search_form:
      search_by_name: Buscar por nombre
      search_by_email: Buscar por email
      search: Buscar
    index:
      follow_user: Seguir usuario
      by_following_a_user: Al seguir un usuario
      search: Buscar
      results: Resultados
    user_search_results:
      no_users_found: No se encontraron usuarios
  cancel_account: Cancelar Cuenta
```

8.2.4. application.rb

```
require File.expand_path('../boot', __FILE__)

require 'rails/all'

if defined?(Bundler)
  # If you precompile assets before deploying to production, use this line
  Bundler.require(*Rails.groups(:assets => %w(development test)))
  # If you want your assets lazily compiled in production, use this line
  # Bundler.require(:default, :assets, Rails.env)
end

module Socialfeed
  class Application < Rails::Application
    # Settings in config/environments/* take precedence over those specified here.
    # Application configuration should go into files in config/initializers
    # -- all .rb files in that directory are automatically loaded.

    # Custom directories with classes and modules you want to be autoloadable.
    # config.autoload_paths += %W(#{config.root}/extras)

    # Only load the plugins named here, in the order given (default is alphabetical).
    # :all can be used as a placeholder for all plugins not explicitly named.
    # config.plugins = [ :exception_notification, :ssl_requirement, :all ]
```

```
    # Activate observers that should always be running.
    # config.active_record.observers = :cacher, :garbage_collector, :forum_observer

    # Set Time.zone default to the specified zone and make Active Record auto-convert to this
zone.
    # Run "rake -D time" for a list of tasks for finding time zone names. Default is UTC.
    # config.time_zone = 'Central Time (US & Canada)'

    # The default locale is :en and all translations from config/locales/*.rb,yml are auto loaded.
    # config.i18n.load_path += Dir[Rails.root.join('my', 'locales', '*.{rb,yml}').to_s]
    config.i18n.default_locale = :es

    # Configure the default encoding used in templates for Ruby 1.9.
    config.encoding = "utf-8"

    # Configure sensitive parameters which will be filtered from the log file.
    config.filter_parameters += [:password]

    # Enable escaping HTML in JSON.
    config.active_support.escape_html_entities_in_json = true

    # Use SQL instead of Active Record's schema dumper when creating the database.
    # This is necessary if your schema can't be completely dumped by the schema dumper,
    # like if you have constraints or database-specific column types
    # config.active_record.schema_format = :sql

    # Enforce whitelist mode for mass assignment.
    # This will create an empty whitelist of attributes available for mass-assignment for all models
    # in your app. As such, your models will need to explicitly whitelist or blacklist accessible
    # parameters by using an attr_accessible or attr_protected declaration.
    config.active_record.whitelist_attributes = true

    # Enable the asset pipeline
    config.assets.enabled = true

    # Services
    config.paths.add File.join('app', 'services'), glob: File.join('**', '*.rb')

    # Version of your assets, change this if you want to expire all your assets
    config.assets.version = '1.0'
  end
end
```

8.2.5. boot.rb

```
require 'rubygems'

# Set up gems listed in the Gemfile.
```

```
ENV['BUNDLE_GEMFILE'] ||= File.expand_path('../../Gemfile', __FILE__)

require 'bundler/setup' if File.exists?(ENV['BUNDLE_GEMFILE'])
```

8.2.6. database.yml

```
# SQLite version 3.x
#   gem install sqlite3
#
#   Ensure the SQLite 3 gem is defined in your Gemfile
#   gem 'sqlite3'
development:
  adapter: sqlite3
  database: db/development.sqlite3
  pool: 5
  timeout: 5000

# Warning: The database defined as "test" will be erased and
# re-generated from your development database when you run "rake".
# Do not set this db to the same as development or production.
test:
  adapter: sqlite3
  database: db/test.sqlite3
  pool: 5
  timeout: 5000

production:
  adapter: sqlite3
  database: db/production.sqlite3
  pool: 5
  timeout: 5000
```

8.2.7. environment.rb

```
# Load the rails application
require File.expand_path('../application', __FILE__)

# Initialize the rails application
Socialfeed::Application.initialize!
```

8.2.8. i18n-tasks.yml

```
# i18n-tasks finds and manages missing and unused translations
https://github.com/glebm/i18n-tasks

base_locale: en
## i18n-tasks detects locales automatically from the existing locale files
## uncomment to set locales explicitly
locales: [es]
```

```
## i18n-tasks report locale, default: en, available: en, ru
# internal_locale: ru

# Read and write locale data
data:
  ## by default, translation data are read from the file system, or you can provide a custom data
adapter
  # adapter: I18n::Tasks::Data::FileSystem

  # Locale files to read from
  read:
    - config/locales/%{locale}.yml
    - config/locales/devise.%{locale}.yml
    # - config/locales/*.%{locale}.yml
    # - config/locales/**/*.%{locale}.yml

  # key => file routes, matched top to bottom
  write:
    ## E.g., write devise and simple form keys to their respective files
    # Catch-all
    - config/locales/%{locale}.yml
    - config/locales/devise.%{locale}.yml
    # `i18n-tasks normalize -p` will force move the keys according to these rules

  # YAML / JSON serializer options, passed to load / dump / parse / serialize
  yaml:
    write:
      # do not wrap lines at 80 characters
      line_width: -1
  json:
    write:
      # pretty print JSON
      indent: '  '
      space: ' '
      object_nl: "\n"
      array_nl: "\n"

# Find translate calls
search:
  ## Default scanner finds t() and I18n.t() calls
  # scanner: I18n::Tasks::Scanners::PatternWithScopeScanner

  ## Paths to search in, passed to File.find
  paths:
    - app/

  ## Root for resolving relative keys (default)
  # relative_roots:
```

```
#   - app/views

  ## File.fnmatch patterns to exclude from search (default)
  # exclude: ["*.jpg", "*.png", "*.gif", "*.svg", "*.ico", "*.eot", "*.ttf", "*.woff", "*.pdf"]

  ## Or, File.fnmatch patterns to include
  # include: ["*.rb", "*.html.slim"]

  ## Lines starting with # or / are ignored by default
  # ignore_lines:
  #  - "^\\s*[#/](?!\\si18n-tasks-use)"

## Google Translate
# translation:
#   # Get an API key and set billing info at https://code.google.com/apis/console to use Google
Translate
#   api_key: "AbC-dEf5"

## Consider these keys not missing
# ignore_missing:
#   - pagination.views.*

## Consider these keys used
# ignore_unused:
#   - 'simple_form.{yes,no}'
#   - 'simple_form.{placeholders,hints,labels}.*'
#   - 'simple_form.{error_notification,required}.:'

## Exclude these keys from `i18n-tasks eq-base' report
# ignore_eq_base:
#   all:
#     - common.ok
#   fr,es:
#     - common.brand

## Exclude these keys from all of the reports
# ignore:
#  - kaminari.*
```

8.2.9. routes.rb

```
Socialfeed::Application.routes.draw do
  # Sidetiq web extension
  require 'sidetiq/web'
  require 'sidekiq/web'
  mount Sidekiq::Web => '/sidekiq'

  devise_for :users, controllers: { registrations: "users/registrations" }
```

```
resources :feeds
resources :entries
resources :categories
resources :filters
resources :notifications
resources :followings, only: [:index, :create, :destroy]

devise_scope :user do
  authenticated :user do
    root to: 'entries#index', as: :authenticated_root
  end

  unauthenticated do
    root to: 'devise/sessions#new', as: :unauthenticated_root
  end
end
# The priority is based upon order of creation:
# first created -> highest priority.

# Sample of regular route:
#   match 'products/:id' => 'catalog#view'
# Keep in mind you can assign values other than :controller and :action

# Sample of named route:
#   match 'products/:id/purchase' => 'catalog#purchase', :as => :purchase
# This route can be invoked with purchase_url(:id => product.id)

# Sample resource route (maps HTTP verbs to controller actions automatically):
#   resources :products

# Sample resource route with options:
#   resources :products do
#     member do
#       get 'short'
#       post 'toggle'
#     end
#
#     collection do
#       get 'sold'
#     end
#   end

# Sample resource route with sub-resources:
#   resources :products do
#     resources :comments, :sales
#     resource :seller
#   end

# Sample resource route with more complex sub-resources
```

```
  #   resources :products do
  #     resources :comments
  #     resources :sales do
  #       get 'recent', :on => :collection
  #     end
  #   end

  # Sample resource route within a namespace:
  #   namespace :admin do
  #     # Directs /admin/products/* to Admin::ProductsController
  #     # (app/controllers/admin/products_controller.rb)
  #     resources :products
  #   end

  # You can have the root of your site routed with "root"
  # just remember to delete public/index.html.

  # See how all your routes lay out with "rake routes"

  # This is a legacy wild controller route that's not recommended for RESTful applications.
  # Note: This route will make all actions in every controller accessible via GET requests.
  # match ':controller(/:action(/:id))(.:format)'
end
```

8.3. db/

8.3.1. migrate/

8.3.1.1. 20140921003303_devise_create_users.rb

```
class DeviseCreateUsers < ActiveRecord::Migration
  def change
    create_table(:users) do |t|
      ## Database authenticatable
      t.string :email,              null: false, default: ""
      t.string :encrypted_password, null: false, default: ""

      ## Recoverable
      t.string   :reset_password_token
      t.datetime :reset_password_sent_at

      ## Rememberable
      t.datetime :remember_created_at

      ## Trackable
      t.integer  :sign_in_count, default: 0, null: false
      t.datetime :current_sign_in_at
      t.datetime :last_sign_in_at
```

```
      t.string   :current_sign_in_ip
      t.string   :last_sign_in_ip

      ## Confirmable
      # t.string   :confirmation_token
      # t.datetime :confirmed_at
      # t.datetime :confirmation_sent_at
      # t.string   :unconfirmed_email # Only if using reconfirmable

      ## Lockable
      # t.integer  :failed_attempts, default: 0, null: false # Only if lock strategy is :failed_attempts
      # t.string   :unlock_token # Only if unlock strategy is :email or :both
      # t.datetime :locked_at


      t.timestamps
    end

    add_index :users, :email,                unique: true
    add_index :users, :reset_password_token, unique: true
    # add_index :users, :confirmation_token,   unique: true
    # add_index :users, :unlock_token,         unique: true
  end
end
```

8.3.1.2. 20141006234754_hobo_migration_1.rb

```
class HoboMigration1 < ActiveRecord::Migration
  def self.up
    add_column :users, :name, :string
  end

  def self.down
    remove_column :users, :name
  end
end
```

8.3.1.3. 20141113024605_add_model_associations.rb

```
class AddModelAssociations < ActiveRecord::Migration
  def self.up
    create_table :categories do |t|
      t.string  :name
      t.string  :tags
      t.text    :description
      t.integer :user_id
    end
    add_index :categories, [:user_id]
```

```
  create_table :entries do |t|
    t.string   :title
    t.string   :url
    t.text     :description
    t.datetime :created_at
    t.datetime :updated_at
    t.integer  :feed_id
  end
  add_index :entries, [:feed_id]

  create_table :filters do |t|
    t.string   :keywords
    t.boolean  :type
    t.boolean  :active
    t.datetime :created_at
    t.datetime :updated_at
    t.integer  :feed_id
  end
  add_index :filters, [:feed_id]

  create_table :notifications do |t|
    t.string  :keywords
    t.integer :feed_id
  end
  add_index :notifications, [:feed_id]

  add_column :users, :notify_new_follower, :boolean, :default => true
  add_column :users, :notify_site_updates, :boolean, :default => true
  add_column :users, :notify_important_topics, :boolean, :default => true

  add_column :feeds, :title, :string
  add_column :feeds, :description, :text
  add_column :feeds, :created_at, :datetime
  add_column :feeds, :updated_at, :datetime
  add_column :feeds, :category_id, :integer

  add_index :feeds, [:category_id]
end

def self.down
  remove_column :users, :notify_new_follower
  remove_column :users, :notify_site_updates
  remove_column :users, :notify_important_topics

  remove_column :feeds, :title
  remove_column :feeds, :description
  remove_column :feeds, :created_at
  remove_column :feeds, :updated_at
  remove_column :feeds, :category_id
```

```
    drop_table :categories
    drop_table :entries
    drop_table :filters
    drop_table :notifications

    remove_index :feeds, :name => :index_feeds_on_category_id rescue
ActiveRecord::StatementInvalid
  end
end
```

8.3.1.4. 20141114025909_add_following_user_self_reference.rb

```
class AddFollowingUserSelfReference < ActiveRecord::Migration
  def self.up
    create_table :followings do |t|
      t.datetime :created_at
      t.datetime :updated_at
      t.integer  :user_id
      t.integer  :followed_id
    end
    add_index :followings, [:user_id]
    add_index :followings, [:followed_id]
  end

  def self.down
    drop_table :followings
  end
end
```

8.3.1.5. 20141116174026_create_saved_entries.rb

```
class CreateSavedEntries < ActiveRecord::Migration
  def change
    create_table :saved_entries do |t|

      t.timestamps
    end
  end
end
```

8.3.1.6. 20141116202630_add_saved_entry_attributes_and_associations.rb

```
class AddSavedEntryAttributesAndAssociations < ActiveRecord::Migration
  def self.up
    drop_table :entries

    add_column :saved_entries, :title, :string
    add_column :saved_entries, :url, :string
```

```
    add_column :saved_entries, :description, :text
    add_column :saved_entries, :feed_id, :integer
    change_column :saved_entries, :created_at, :datetime, :null => true
    change_column :saved_entries, :updated_at, :datetime, :null => true

    add_column :categories, :created_at, :datetime
    add_column :categories, :updated_at, :datetime

    add_column :notifications, :created_at, :datetime
    add_column :notifications, :updated_at, :datetime

    add_index :saved_entries, [:feed_id]
  end

  def self.down
    remove_column :saved_entries, :title
    remove_column :saved_entries, :url
    remove_column :saved_entries, :description
    remove_column :saved_entries, :feed_id
    change_column :saved_entries, :created_at, :datetime, :null => false
    change_column :saved_entries, :updated_at, :datetime, :null => false

    remove_column :categories, :created_at
    remove_column :categories, :updated_at

    remove_column :notifications, :created_at
    remove_column :notifications, :updated_at

    create_table "entries", :force => true do |t|
      t.string   "title"
      t.string   "url"
      t.text     "description"
      t.datetime "created_at"
      t.datetime "updated_at"
      t.integer  "feed_id"
    end

    add_index "entries", ["feed_id"], :name => "index_entries_on_feed_id"

    remove_index :saved_entries, :name => :index_saved_entries_on_feed_id rescue
ActiveRecord::StatementInvalid
  end
end
```

8.3.1.7. 20141117000241_rename_to_channel_and_saved_article.rb

```
class RenameToChannelAndSavedArticle < ActiveRecord::Migration
  def self.up
    rename_table :feeds, :channels
```

```
    rename_table :saved_entries, :saved_articles

    rename_column :saved_articles, :feed_id, :channel_id

    rename_column :filters, :feed_id, :channel_id

    rename_column :notifications, :feed_id, :channel_id

    remove_index :channels, :name => :index_feeds_on_category_id rescue
ActiveRecord::StatementInvalid
    add_index :channels, [:category_id]

    remove_index :saved_articles, :name => :index_saved_entries_on_feed_id rescue
ActiveRecord::StatementInvalid
    add_index :saved_articles, [:channel_id]

    remove_index :filters, :name => :index_filters_on_feed_id rescue
ActiveRecord::StatementInvalid
    add_index :filters, [:channel_id]

    remove_index :notifications, :name => :index_notifications_on_feed_id rescue
ActiveRecord::StatementInvalid
    add_index :notifications, [:channel_id]
  end

  def self.down
    rename_column :saved_articles, :channel_id, :feed_id

    rename_column :filters, :channel_id, :feed_id

    rename_column :notifications, :channel_id, :feed_id

    rename_table :channels, :feeds
    rename_table :saved_articles, :saved_entries

    remove_index :feeds, :name => :index_channels_on_category_id rescue
ActiveRecord::StatementInvalid
    add_index :feeds, [:category_id]

    remove_index :saved_entries, :name => :index_saved_articles_on_channel_id rescue
ActiveRecord::StatementInvalid
    add_index :saved_entries, [:feed_id]

    remove_index :filters, :name => :index_filters_on_channel_id rescue
ActiveRecord::StatementInvalid
    add_index :filters, [:feed_id]

    remove_index :notifications, :name => :index_notifications_on_channel_id rescue
ActiveRecord::StatementInvalid
```

```
    add_index :notifications, [:feed_id]
  end
end
```

8.3.1.8. 20141117233711_create_entries.rb

```
class CreateEntries < ActiveRecord::Migration
  def change
    create_table :entries do |t|

      t.timestamps
    end
  end
end
```

8.3.1.9. 20141117235846_rename_back_to_feeds_and_entries.rb

```
class RenameBackToFeedsAndEntries < ActiveRecord::Migration
  def self.up
    rename_table :channels, :feeds

    drop_table :saved_articles

    add_column :entries, :name, :string
    add_column :entries, :summary, :text
    add_column :entries, :url, :string
    add_column :entries, :published_at, :datetime
    add_column :entries, :guid, :string
    add_column :entries, :feed_id, :integer
    change_column :entries, :created_at, :datetime, :null => true
    change_column :entries, :updated_at, :datetime, :null => true

    rename_column :filters, :channel_id, :feed_id

    rename_column :notifications, :channel_id, :feed_id

    add_index :entries, [:feed_id]

    remove_index :feeds, :name => :index_channels_on_category_id rescue
ActiveRecord::StatementInvalid
    add_index :feeds, [:category_id]

    remove_index :filters, :name => :index_filters_on_channel_id rescue
ActiveRecord::StatementInvalid
    add_index :filters, [:feed_id]

    remove_index :notifications, :name => :index_notifications_on_channel_id rescue
ActiveRecord::StatementInvalid
    add_index :notifications, [:feed_id]
```

```
  end

  def self.down
    remove_column :entries, :name
    remove_column :entries, :summary
    remove_column :entries, :url
    remove_column :entries, :published_at
    remove_column :entries, :guid
    remove_column :entries, :feed_id
    change_column :entries, :created_at, :datetime, :null => false
    change_column :entries, :updated_at, :datetime, :null => false

    rename_column :filters, :feed_id, :channel_id

    rename_column :notifications, :feed_id, :channel_id

    rename_table :feeds, :channels

    create_table "saved_articles", :force => true do |t|
      t.datetime "created_at"
      t.datetime "updated_at"
      t.string   "title"
      t.string   "url"
      t.text     "description"
      t.integer  "channel_id"
    end

    add_index "saved_articles", ["channel_id"], :name => "index_saved_articles_on_channel_id"

    remove_index :entries, :name => :index_entries_on_feed_id rescue
ActiveRecord::StatementInvalid

    remove_index :channels, :name => :index_feeds_on_category_id rescue
ActiveRecord::StatementInvalid
    add_index :channels, [:category_id]

    remove_index :filters, :name => :index_filters_on_feed_id rescue
ActiveRecord::StatementInvalid
    add_index :filters, [:channel_id]

    remove_index :notifications, :name => :index_notifications_on_feed_id rescue
ActiveRecord::StatementInvalid
    add_index :notifications, [:channel_id]
  end
end
```

8.3.1.10. 20141118000247_rename_feeds_title_to_name.rb

```
class RenameFeedsTitleToName < ActiveRecord::Migration
```

```
  def self.up
    rename_column :feeds, :title, :name
  end

  def self.down
    rename_column :feeds, :name, :title
  end
end
```

8.3.1.11. 20141119005144_add_etag_to_feed.rb

```
class AddEtagToFeed < ActiveRecord::Migration
  def self.up
    add_column :feeds, :etag, :string
  end

  def self.down
    remove_column :feeds, :etag
  end
end
```

8.3.1.12. 20141119030809_add_removable_attribute_to_category.rb

```
class AddRemovableAttributeToCategory < ActiveRecord::Migration
  def self.up
    add_column :categories, :removable, :boolean, :default => true
  end

  def self.down
    remove_column :categories, :removable
  end
end
```

8.3.1.13. 20141119031937_drop_removable_add_editable_to_category.rb

```
class DropRemovableAddEditableToCategory < ActiveRecord::Migration
  def self.up
    add_column :categories, :editable, :boolean, :default => true
    remove_column :categories, :removable
  end

  def self.down
    remove_column :categories, :editable
    add_column :categories, :removable, :boolean, :default => true
  end
end
```

8.3.1.14. 20141120005149_add_last_modified_to_feed.rb

```
class AddLastModifiedToFeed < ActiveRecord::Migration
  def self.up
    add_column :feeds, :last_modified, :datetime
  end

  def self.down
    remove_column :feeds, :last_modified
  end
end
```

8.3.1.15. 20150107175743_add_favorite_to_entries.rb

```
class AddFavoriteToEntries < ActiveRecord::Migration
  def self.up
    add_column :entries, :favorite, :boolean
  end

  def self.down
    remove_column :entries, :favorite
  end
end
```

8.3.1.16. 20150108152420_add_read_to_entries.rb

```
class AddReadToEntries < ActiveRecord::Migration
  def self.up
    add_column :entries, :read, :boolean
  end

  def self.down
    remove_column :entries, :read
  end
end
```

8.3.1.17. 20150113172738_add_user_id_for_tenancy.rb

```
class AddUserIdForTenancy < ActiveRecord::Migration
  def self.up
    add_column :entries, :user_id, :integer

    add_column :feeds, :user_id, :integer

    add_column :filters, :user_id, :integer

    add_column :notifications, :user_id, :integer
  end

  def self.down
    remove_column :entries, :user_id
```

```
    remove_column :feeds, :user_id

    remove_column :filters, :user_id

    remove_column :notifications, :user_id
  end
end
```

8.3.1.18. 20150122190508_set_filter_defaults.rb

```
class SetFilterDefaults < ActiveRecord::Migration
  def self.up
    change_column :filters, :type, :boolean, :default => false
    change_column :filters, :active, :boolean, :default => true
  end

  def self.down
    change_column :filters, :type, :boolean
    change_column :filters, :active, :boolean
  end
end
```

8.3.1.19. 20150123175233_change_filters_type_to_list_type.rb

```
class ChangeFiltersTypeToListType < ActiveRecord::Migration
  def self.up
    rename_column :filters, :type, :list_type
  end

  def self.down
    rename_column :filters, :list_type, :type
  end
end
```

8.3.1.20. 20150128201327_add_active_field_to_notification.rb

```
class AddActiveFieldToNotification < ActiveRecord::Migration
  def self.up
    add_column :notifications, :active, :boolean, :default => true
  end

  def self.down
    remove_column :notifications, :active
  end
end
```

8.3.1.21. 20150129222859_add_content_to_entry.rb

```
class AddContentToEntry < ActiveRecord::Migration
  def self.up
    add_column :entries, :content, :text
  end

  def self.down
    remove_column :entries, :content
  end
end
```

8.3.2. schema.rb

```
# encoding: UTF-8
# This file is auto-generated from the current state of the database. Instead
# of editing this file, please use the migrations feature of Active Record to
# incrementally modify your database, and then regenerate this schema definition.
#
# Note that this schema.rb definition is the authoritative source for your
# database schema. If you need to create the application database on another
# system, you should be using db:schema:load, not running all the migrations
# from scratch. The latter is a flawed and unsustainable approach (the more migrations
# you'll amass, the slower it'll run and the greater likelihood for issues).
#
# It's strongly recommended to check this file into your version control system.

ActiveRecord::Schema.define(:version => 20150129222859) do

  create_table "categories", :force => true do |t|
    t.string   "name"
    t.string   "tags"
    t.text     "description"
    t.integer  "user_id"
    t.datetime "created_at"
    t.datetime "updated_at"
    t.boolean  "editable",    :default => true
  end

  add_index "categories", ["user_id"], :name => "index_categories_on_user_id"

  create_table "entries", :force => true do |t|
    t.datetime "created_at"
    t.datetime "updated_at"
    t.string   "name"
    t.text     "summary"
    t.string   "url"
    t.datetime "published_at"
    t.string   "guid"
    t.integer  "feed_id"
    t.boolean  "favorite",     :default => false
```

```
  t.boolean  "read",         :default => false
  t.integer  "user_id"
  t.text     "content"
end

add_index "entries", ["feed_id"], :name => "index_entries_on_feed_id"

create_table "feeds", :force => true do |t|
  t.string   "url"
  t.string   "name"
  t.text     "description"
  t.datetime "created_at"
  t.datetime "updated_at"
  t.integer  "category_id"
  t.string   "etag"
  t.datetime "last_modified"
  t.integer  "user_id"
end

add_index "feeds", ["category_id"], :name => "index_feeds_on_category_id"

create_table "filters", :force => true do |t|
  t.string   "keywords"
  t.boolean  "list_type",  :default => false
  t.boolean  "active",     :default => true
  t.datetime "created_at"
  t.datetime "updated_at"
  t.integer  "feed_id"
  t.integer  "user_id"
end

add_index "filters", ["feed_id"], :name => "index_filters_on_feed_id"

create_table "followings", :force => true do |t|
  t.datetime "created_at"
  t.datetime "updated_at"
  t.integer  "user_id"
  t.integer  "followed_id"
end

add_index "followings", ["followed_id"], :name => "index_followings_on_followed_id"
add_index "followings", ["user_id"], :name => "index_followings_on_user_id"

create_table "notifications", :force => true do |t|
  t.string   "keywords"
  t.integer  "feed_id"
  t.datetime "created_at"
  t.datetime "updated_at"
  t.integer  "user_id"
```

```
    t.boolean  "active",     :default => true
  end

  add_index "notifications", ["feed_id"], :name => "index_notifications_on_feed_id"

  create_table "users", :force => true do |t|
    t.string   "email",                  :default => "",   :null => false
    t.string   "encrypted_password",     :default => "",   :null => false
    t.string   "reset_password_token"
    t.datetime "reset_password_sent_at"
    t.datetime "remember_created_at"
    t.integer  "sign_in_count",          :default => 0,    :null => false
    t.datetime "current_sign_in_at"
    t.datetime "last_sign_in_at"
    t.string   "current_sign_in_ip"
    t.string   "last_sign_in_ip"
    t.datetime "created_at",                               :null => false
    t.datetime "updated_at",                               :null => false
    t.string   "name"
    t.boolean  "notify_new_follower",    :default => true
    t.boolean  "notify_site_updates",    :default => true
    t.boolean  "notify_important_topics", :default => true
  end

  add_index "users", ["email"], :name => "index_users_on_email", :unique => true
  add_index "users", ["reset_password_token"], :name =>
"index_users_on_reset_password_token", :unique => true

end
```

8.4. lib/

8.4.1. tasks/

8.4.1.1. test_data.rake

```
namespace :demo do
  desc <<-DESC
    Load testing data.
    Run using the command 'rake demo:load_data'
  DESC

  task :reset => [:environment] do
    Rake::Task["db:drop"].invoke
    Rake::Task["db:create"].invoke
    Rake::Task["db:migrate"].invoke
  end
```

```
  task :load_all => [:environment] do
    puts "Loading All"
    Rake::Task["demo:create_users"].execute
    Rake::Task["demo:create_categories"].execute
    Rake::Task["demo:create_feeds"].execute
  end

  task :destroy_all => [:environment] do
    puts "Destroying All"
    [User, Category, Feed, Filter, Notification].each(&:destroy_all)
  end

  task :create_users => [:environment] do
    puts "Creating Users"
    User.create!(name: 'Pedro Rodríguez', email: 'pepe@pepe.com', password: 'pepepepe') unless
User.exists? email: 'pepe@pepe.com'
    User.create!(name: 'José Fernández', email: 'jose@jose.com', password: 'josejose') unless
User.exists? email: 'jose@jose.com'
    User.create!(name: 'Daniela Guzmán', email: 'raul@raul.com', password: 'raulraul') unless
User.exists? email: 'raul@raul.com'
    User.create!(name: 'Julio Soroeta', email: 'manuel@manuel.com', password: 'manuelmanuel')
unless User.exists? email: 'manuel@manuel.com'
  end

  task :create_categories => [:environment] do
    puts "Creating Categories"
    Category.create!(name: 'News', description: 'Politics, Economy and Entertainment', user_id:
User.where(email: 'jose@jose.com').first.id) unless Category.exists? name: 'News', user_id:
User.where(email: 'jose@jose.com').first.id
    Category.create!(name: 'Sports', description: 'Football, Volleyball and Basketball', user_id:
User.where(email: 'pepe@pepe.com').first.id) unless Category.exists? name: 'Sports', user_id:
User.where(email: 'pepe@pepe.com').first.id
    Category.create!(name: 'Technology', description: 'Games, Mobile and Web applications news',
user_id: User.where(email: 'pepe@pepe.com').first.id) unless Category.exists? name:
'Technology', user_id: User.where(email: 'pepe@pepe.com').first.id
  end

  task :create_feeds => [:environment] do
    puts "Creating Feeds"
    Feed.create!(name: 'News', description: 'Politics, Economy and Entertainment', url:
'http://www.theguardian.com/uk/rss', category_id: Category.where(name: 'News').first.id) unless
Feed.exists? name: 'News'
    Feed.create!(name: 'Sports', description: 'Football, Volleyball and Basketball', url:
'https://sports.yahoo.com/blogs/rss.xml', category_id: Category.where(name: 'Sports').first.id)
unless Feed.exists? name: 'Sports'
    Feed.create!(name: 'Technology', description: 'Noticias de Internet y Tecnología', url:
'http://www.clarin.com/rss/internet/', category_id: Category.where(name: 'Technology').first.id)
unless Feed.exists? name: 'Technology'
  end
```

```
  task :create_filters => [:environment] do
    # puts "Creating Filters"
    # Filter.create! title: 'News', description: 'Politics, Economy and Entertainment', url: 'someurl'
  end

  task :create_notifications => [:environment] do
    # puts "Creating Notifications"
    # Notification.create! title: 'News', description: 'Politics, Economy and Entertainment', url:
'someurl'
  end
end
```

8.5. .gitignore

```
# See http://help.github.com/ignore-files/ for more about ignoring files.
#
# If you find yourself ignoring temporary files generated by your text editor
# or operating system, you probably want to add a global ignore instead:
#   git config --global core.excludesfile ~/.gitignore_global

# Ignore bundler config
/.bundle

# Ignore the default SQLite database.
/db/*.sqlite3

# Ignore all logfiles and tempfiles.
/log/*.log
/tmp
dump.rdb
```

8.6. Gemfile

```
source 'https://rubygems.org'

gem 'rails', '3.2.16'

# Bundle edge Rails instead:
# gem 'rails', :git => 'git://github.com/rails/rails.git'

# Database
gem 'sqlite3'

# Feeds
gem 'feedjira'

# Background Processing
```

```
gem 'sinatra', require: nil
gem 'sidekiq'
gem 'sidetiq'

# DB Migrations
gem 'hobo_fields'

# Forms
gem 'simple_form'

# Select boxes
gem "select2-rails"

# HAML
gem 'haml-rails'

# Decorators
gem 'draper'

# Authentication
gem 'devise'

# Multi-tenancy
gem 'acts_as_tenant'

# File Upload
gem 'carrierwave'

# Internationalization
gem 'i18n-tasks'

# Pagination
gem 'kaminari'

# Searches
gem 'polyamorous'
gem 'ransack'

# Input tags
gem 'bootstrap-tagsinput-rails'

group :development do
        gem "erb2haml"
  # Errors
  gem 'better_errors'
  gem 'binding_of_caller'
  # Debugger
  gem 'pry'
  gem 'pry-rails'
```

```
  gem 'pry-byebug' unless Gem.win_platform?
end

# Testing
group :test, :development do
        gem 'rspec-rails'
        gem 'cucumber-rails', require: false
        gem 'factory_girl_rails'
  gem 'letter_opener'
end

# Backend
# gem 'activeadmin'

# Gems used only for assets and not required
# in production environments by default.
group :assets do
  gem 'sass-rails'
  gem 'sass'
  gem 'bootstrap-sass'

  gem 'uglifier', '>= 1.0.3'
  gem 'coffee-rails'
  # Front-End framework
  gem 'therubyracer', :platforms => :ruby
  gem 'twitter-bootstrap-rails', :git => 'git://github.com/seyhunak/twitter-bootstrap-rails.git'
  gem 'autoprefixer-rails'
end

gem 'jquery-rails'

# To use ActiveModel has_secure_password
# gem 'bcrypt-ruby', '~> 3.0.0'

# To use Jbuilder templates for JSON
# gem 'jbuilder'

# Use unicorn as the app server
# gem 'unicorn'

# Deploy with Capistrano
# gem 'capistrano'

# To use debugger
# gem 'debugger'
```

8.7. Procfile.dev

```
zeus:    zeus start
```

```
redis:      redis-server
sidekiq: bundle exec sidekiq
```

8.8. run.sh

```
rm .zeus.sock
killall -9 redis-server
killall -9 zeus-linux-386
killall -9 ruby
killall -9 inotify-wrapper
foreman start -f Procfile.dev
```