Universidad de Valladolid

E.T.S Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Desarrollo SaaS con Ruby on Rails

Autor:
Alfonso Peralta Calvo

Universidad de Valladolid

E.T.S Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Desarrollo SaaS con Ruby on Rails

Autor:
Alfonso Peralta Calvo
Tutora:
Yania Crespo González-Carvajal

Resumen

Con el objetivo de liberar a los dispositivos finales de lógica de cómputo, muchas veces esta se traslada al servidor y estos terminales simplemente se tienen que ocupar de consultar información. Las aplicaciones web se alojan en la nube y hacen el desarrollo y el mantenimiento más sencillos.

En este Trabajo de Fin de Grado se ha implementado una aplicación web accesible por navegador y desarrollada con metodologías ágiles. Mediante una serie de herramientas y ténicas de desarrollo se ha cubierto un desarrollo de un proyecto software en todas sus etapas hasta la producción de una versión definitiva. En este documento se especifica tanto las herramientas y técnicas utilizadas como la evolución del desarrollo del proyecto en sus diferentes etapas.

Índice

Índice de figuras

1 Introducción

1.1. Contexto

En los últimos años hemos visto un aumento de dispositivos portátiles o móviles de uso cotidiano. Muchas veces estos dispositivos deben llevar a cabo procesamientos costosos y tratamiento de datos masivo.

Existe la tendencia de liberar a los dispositivos de funciones complejas y alojarlas en un servidor de la empresa de tal manera que sean ellos los que se ocupen del mantenimiento, operación diaria y soporte de los procesamientos, mientras que el dispositivo se convierte en un aparato de consulta.

Situando la computación y la lógica de la aplicación en un servidor conectado a internet resolvemos el problema, además el mantenimiento del software se simplifica, teniendo que actualizarse y mantenerse en un solo sitio (el servidor) en vez de en múltiples versiones distribuidas en los dispositivos. Este tipo de aplicaciones se engloban bajo el término Software como Servicio (Software as a Service, SaaS).

1.2. Objetivos

En este trabajo de fin de grado se ha tratado de afrontar el problema del desarrollo de una aplicación SaaS, comenzando por una planificación basada en métodos ágiles y terminando en la elección e implementación de un framework de alto nivel tecnológico. Como mecanismo de interacción con el usuario se ha optado por emplear una aplicación web accesible desde cualquier navegador y que permita el uso de las funciones que la aplicación ofrece. Los objetivos de este trabajo son los siguientes:

- Planificación de un proyecto software usando métología ágil, desarrollo basado en test (TDD) y desarrollo basado en comportamiento (BDD).
- Elaboración de prototipos y demos funcionales tempranas.
- Elección, aprendizaje e implementación de un framework avanzado (Ruby on Rails).

1.3. Movitación

Este proyecto no surgió como una propuesta de TFG en las listas ofertadas en Octubre, sino que fue un proceso de iniciativa y diálogo profesor-alumno lo que lo originó.

Mi propuesta inicial a la profesora Yania Crespo González-Carvajal fue la de hacer algún tipo de proyecto en Ruby on Rails. Esta idea surgió de la asignatura Diseño de Software, en la que una de las tareas era realizar un estudio sobre un framework avanzado. En mi caso la elección fue Ruby on Rails. El hecho de tener que investigar sobre este framework, alimentó mi curiosidad y vi el TFG como una oportunidad para llevarlo a cabo

Además Yania Crespo me comentó que ya había desarrollado algún proyecto de fin de carrera en Ruby on Rails, otro punto a favor.

He visto este proyecto como una oportunidad para aprender un nuevo framework de desarrollo web avanzado, y de utilizar una metodología de desarrollo no tradicional (ágil) en vez de implementar algúna aplicación en lenguajes, tecnología y metodología conocidos.

1.4. Estructura de la memoria

En esta memoria se recoge desde una explicación de las metodologías y tecnologías empleadas hasta el desarrollo del proyecto.

- **Marco teórico:** En este apartado se han recogido los métodos empleados para plantear, planificar y desarrollar el proyecto. También se argumenta por qué esta metodología es adecuada para nuestro problema en concreto.
- **Marco tecnológico:** Incluye una breve explicación de qué trata cada tecnología (lenguajes y herramientas) empleada para el desarrollo e implementación del proyecto.
- **Proyecto software:** Se explica cómo se ha decidido desarrollar el proyecto, cómo se han planteado las reuniones de toma de requisitos, cómo ha quedado el proyecto tras cada iteración y el resultado final del mismo.
- **Bibliografía:** Referencias bibliográficas del proyecto.
- **Anexos:** Se añaden al final las imágenes o elementos adicionales.

1.5. Referencia

Como referencia hemos tomado un libro escrito por Armando Fox y David Patterson en el año 2012 titulado Engineering Software as a Service: An Agile Approach Using Cloud Computing [1] y que trata de explicar el desarrollo SaaS, la metodología ágil y por qué es adecuado emplear esta metodología en este tipo de problemas. Además el libro sirve de complemento a unos cursos MOOC (Massively Open Online Course) de la universidad de Berkeley CS169.1x [2] y CS169.2x [3].

2 Marco Teórico

En este capítulo vamos a explicar la metodología que hemos elegido y empleado para desarrollar desde el principio un proyecto software. Los métodos y procesos de desarrollo utilizados entran dentro de lo que se conoce como métodos ágiles.

2.1. Desarrollo Ágil

2.1.1. Introducción

Los métodos ágiles refieren a métodos iterativos e incrementales de ingeniería de software. Estos métodos hacen especial énfasis en una comunicación directa y continua con el cliente y en hacer primar estas comunicaciones sobre la documentación escrita. Las ideas en las que se basan estos métodos fueron resumidas en el Manifiesto Ágil [4]. En general estos métodos promueven:

- Planificación con capacidad de adaptación.
- Desarrollo evolutivo.
- Comunicación con el cliente sobre documentación.
- Entrega temprana de demos funcionales.
- Mejora contínua [5].
- Respuesta flexible a los cambios.

2.1.2. Sprints

Las entregas se realizan después de cada iteración o periodo de desarrollo, este concepto se conoce como Sprints. Estos periodos suelen tener una duración de dos semanas en las que desarrollamos una serie de features (funcionalidades) que hemos planificado previamente. La idea es tener un prototipo o demo funcional que integre los cambios añadidos en las features.

Al tener periodos de desarrollo tan cortos, conseguimos minimizar el riesgo de desarrollar algo incorrectamente. Tras cada periodo de desarrollo se debe llevar a cabo una reunión con el cliente para verificar los cambios implementados y establecer nuevos requisitos mediante historias de usuario.

2.1.3. Historias de usuario

Las historias de usuario o User Stories son representaciones de requisitos expresados en un lenguaje común entendible por el usuario. Tienen la capacidad de responder rápidamente a requisitos cambiantes y evitan gran cantidad de documentos formales.

Es el equivalente en metodología ágil al paso de toma de requisitos a casos de uso. Son un acuerdo que formaliza la funcionalidad descrita y esperada de una parte de la aplicación. La historia de usuario incluye las pruebas de aceptación escritas en un lenguaje formal (ver 3.5, Capybara y Cucumber,).

La descripción o enunciado de la historia de usuario tiene una forma de ser construida, con la estructura “Como - Quiero - Para”, en inglés “As a - I want to - So that”:

- **Como:** Describimos el rol que tomamos en esta historia y/o las condiciones que se dan para ese rol. (ej: *Como usuario logeado*)

- **Quiero:** Describimos expresamente qué queremos que ocurra en la historia. (ej: *Quiero ver un icono de un sobre.*)

- **Para:** Aquí ponemos el objetivo que buscamos cumplir con la historia (ej: *Para poder hacer click y mandar un email en cualquier momento.*)

Es común el uso de historias de usuario en tarjetas físicas, post-it o pizarras electrónicas que permitan la visualización de estas para poder plantear un orden o prioridades. En cada iteración o periodo de desarrollo se escogen una serie de historias a las que asignamos un peso o dificultad y una prioridad de manera que se desarrollen antes las historias con mayor prioridad.

Estas historias deben tener una serie de características:

- **Independientes:** cada una tiene un resultado independiente del resto

- **Negociables:** La discusión con los usuarios debe permitir esclarecer su alcance y éste debe dejarse explícito bajo la forma de pruebas de validación.

- **Valoradas por los clientes:** Cada historia debe ser importante para los usuarios más que para el desarrollador.

- **Estimables:** Un resultado de la discusión de una historia de usuario es la estimación del tiempo que tomará completarla.

- **Short:** Generalmente se recomienda la consolidación de historias muy cortas en una sola historia.

- **Testeable:** Las historias de usuario cubren requerimientos funcionales, por lo que generalmente son verificables.

2.1.4. Ventajas y Desventajas

- **Ventajas**

  - Buena respuesta a los cambios en los requistos [6]
  - Al tener trato continuo con el cliente no hay lugar para suposiciones (de requisitos).
  - El equipo no tiene que preocuparse de que cuando entreguen el producto, los requisitos hayan cambiado.

- **Desventajas**

  - Es complicado calcular el esfuerzo total requerido al principio del proyecto.
  - Depende de que el cliente exprese los requisitos correctamente.
  - Falta de énfasis en diseños y documentación a veces necesarios.

2.1.5. ¿Por qué usar desarrollo ágil?

Como ya hemos explicado, hemos usado el libro Engineering Software as a Service: An Agile Approach Using Cloud Computing [1] como referencia para el desarrollo del proyecto. La segunda mitad de este libro está dedicada a metodología ágil y tipos de desarrollo dentro de este marco. El desarrollo ágil se complementa perfectamente con las tecnologías escogidas para implementar el proyecto, pues podemos prototipar muy fácilmente y desde muy temprano. Esto hace que el cliente sea plenamente consciente del estado real de la aplicación tras cada iteración o periodo de desarrollo.

2.2. TDD. Desarrollo Basado en Test

2.2.1. Introducción

Las siglas TDD vienen de Test Driven Development. Al contrario de la programación tradicional, donde primero se programa y luego se prueba lo que se ha escrito, en TDD primero se escriben los test. Esta filosofía hace que primero pensemos una cosa concreta que debe hacer nuestra aplicación, luego hacemos el test que prueba que nuestra aplicación hace eso y después escribimos el código que valida ese test. No se hace nada de código si no falla ningún test, por lo que primero tenemos que tener tests que fallen. Por tanto los pasos serían los siguientes:

1. Hacer un test automático de prueba, ejecutarlo y ver que falla.
2. Hacer el código mínimo imprescindible para que el test que acabamos de escribir pase.
3. Refactorizar, sobre todo, para evitar cosas duplicadas en el mismo.
4. Volver a pasar el test tras la refactorización.

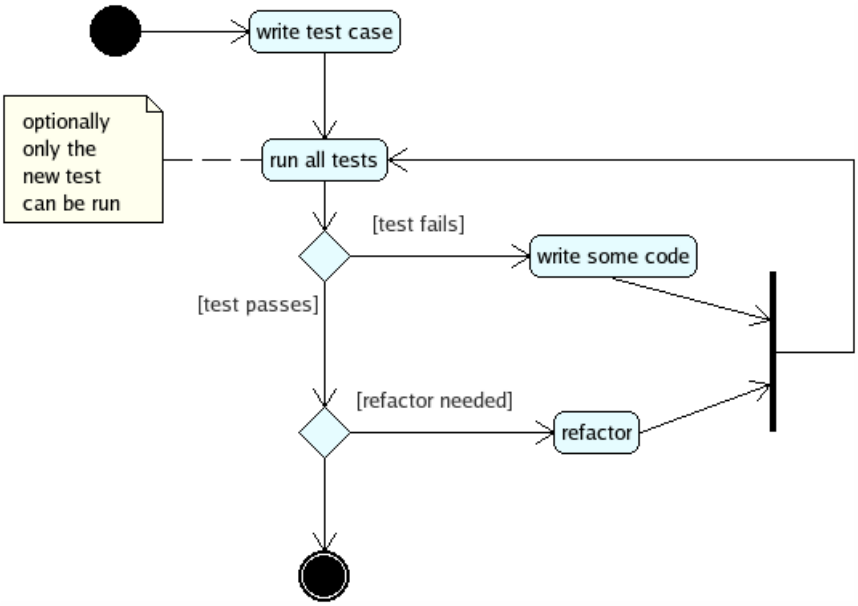

Figura 1: Flujo de desarrollo usando TDD [7]

En la Figura 1 podemos observar la sucesión de operaciones que se dan en un desarrollo basado en test. Por esta secuencia de pasos, el desarrollo ágil se denomina alternativamente ciclo Rojo-Verde-Refactor, atendiendo a los colores rojo asociado al fallo del test y verde asociado a test que se ejecutan con éxito.

2.2.2. Ventajas y desventajas

- **Ventajas**
  - Menor cantidad de código, solamente se escribe hasta que se satisface un test.
  - La refactorización de código forma parte del proceso.

- Requiere menor tiempo de debug.

- **Desventajas**

  - Necesidad de tiempo de adaptación a este tipo de desarrollo.
  - Realizar test de casos complejos puede resultar en mayor tiempo total.
  - Los test pueden verse afectados si se hacen cambios en el diseño.

2.2.3. ¿Por qué usar TDD?

De nuevo, el capítulo 8 de Engineering Software as a Service: An Agile Approach Using Cloud Computing [1] trata de TDD. El desarrollo basado en test es especialmente útil al tratar con casos unitarios, si se tratan de test de integración o más amplios quizá habría que pensar en usar BDD. La elección de usar TDD o programar algo de la manera convencional, es en cierta manera subjetiva. Lo que nos asgura TDD es que si seguimos el procedimiento acabaremos con ún código de cierta calidad y sin funcionalidades innecesarias.

2.3. BDD. Desarrollo Basado en Comportamiento

2.3.1. Introducción

El Desarrollo Basado en Comportamiento o BDD (Behaviour Driver Development) se puede entender como una evolución del TDD añadiendo elementos propios de la orientación a objeto [8]. Mantiene la manera de desarrollar de TDD (test-first) pero estos test se intentan visualizar desde la perspectiva del usuario y el comportamiento de este.
Este tipo de desarrollo busca crear un marco de entendimiento común entre negocio y tecnología, entre cliente y desarrollador. Este marco está soportado por un lenguaje común en el que especificar los requisitos de manera que puedan ser verificados por el cliente de manera sencilla [9]. De esta manera eliminamos el paso de toma de requisitos a diseño en el que podemos perder ciertas especificaciones no tan claras.

2.3.2. Ventajas y desventajas

- **Ventajas**
  - Hace énfasis en el comportamiento en vez de saltar a pensar en cómo implementar.
  - Refuerza el trato y acomoda la comunicación con el cliente.
  - Hace especial énfasis en entender el funcionamiento del negocio. Esto hace que podamos dar prioridad a características en función del valor de negocio que aporten.
  - Los test de comportamiento actúan en parte como documentación.
- **Desventajas**
  - Es complicado convencer al cliente de que tiene que contribuir a construir los escenarios de comportamiento.
  - El cliente colabora en los escenarios, lo que puede llevar a comportamientos que a priori parecen adecuados pero a la hora de la implementación no lo son.
  - Se necesita un tiempo de adaptación a este lenguaje común cliente-desarrollador.

2.3.3. ¿Por qué usar BDD?

Una vez hemos decidido emplear metodología ágil en la planificación y TDD en la pruebas unitarias, el uso del desarrollo basado en comportamiento para las features es el complemento perfecto.

Establecemos un escenario, diseñamos unas pruebas con el cliente en el lenguaje común que hemos establecido, y a la hora de implementar utilizamos las pruebas unitarias de TDD para generar el código hasta que satisfacemos el escenario. En la Figura 2 podemos observar cómo estos pasos se suceden.

El capítulo 7 de Engineering Software as a Service: An Agile Approach Using Cloud Computing [1] versa sobre BDD, por qué es adecuado y presenta unas herramientas con las que implementar los test de comportamiento.

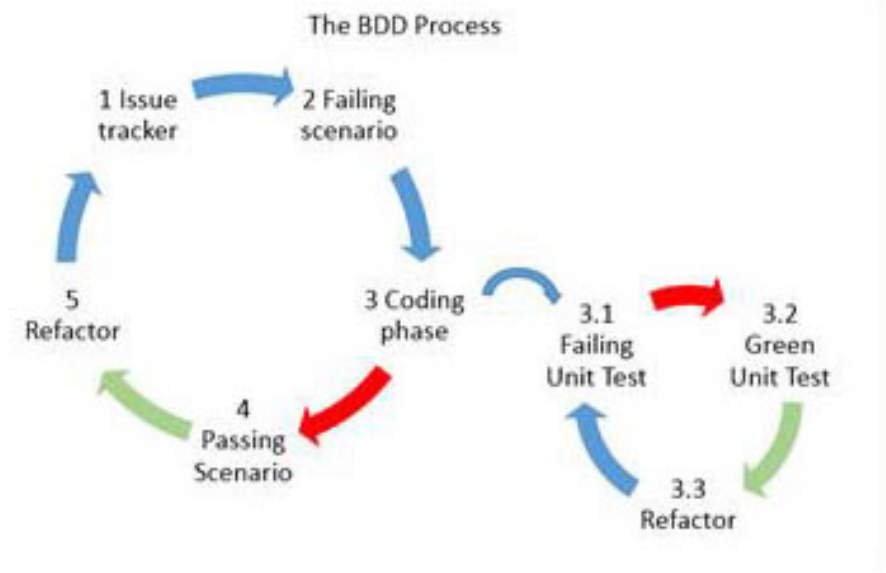

Figura 2: Proceso de desarrollo de un proyecto BDD

3 Marco Tecnológico

En este capítulo explicamos con detalle cada una de las tecnologías o herramientas de las que hemos hecho uso en el transcurso del desarrollo del TFG. Se van a introducir en orden, primero los lenguajes de programación, luego el framework, lenguajes de diseño y base de datos y por último otras tecnologías que nos han ayudado a planificarlo de manera ágil.

3.1. Ruby

Ruby es un lenguaje de programación de alto nivel. Es interpretado (scripted), orientado a objetos y reflexivo, también soporta el paradigma funcional y el imperativo. Fue creado en 1995 por Yukihiro "Matz" Matsumoto con la intención de que fuera una simplificación de Lisp, con una gestión de la orientación a objetos similar a la de Smalltalk y la utilidad práctica de un lenguaje como Perl [10].

3.1.1. Todo en Ruby es un objeto

Una de las cosas que Matsumoto pretendía solucionar cuando creó este lenguaje era hacer un lenguaje con mayor orientación a objeto que Python [11]. En Ruby todo es visto como un objeto. Toda porción de información posee sus propias acciones y propiedades [12]. Estas propiedades, Ruby las denomina "variables de instancia" y a las acciones "métodos".

Por ejemplo, un número entero sería una objeto de clase Fixnum que implementa gran cantidad de métodos con lo que sería posible escribir el siguiente código:

```
5.times{ print " Hola Mundo..." }
```

Cuya salida sería:

```
Hola Mundo... Hola Mundo... Hola Mundo... Hola Mundo... Hola Mundo...
```

En muchos lenguajes, los números u otros tipos primitivos no son tratados como objetos. Ruby sigue la influencia de Smalltalk para proporcionar métodos y variables de instancia a todos los tipos.

3.1.2. Metaprogramación

Una de las características más potentes de Ruby es la metaprogramación. Entendemos como metaprogramación la capacidad de un programa de modificar su propio código. Como ejemplo podemos hacer un método de una clase que se ejecute cuando llames a cualquier método que no exista.

```
class Helado
  def method_missing(method, *args)
    attribute = name.to_s
    @attributes[attribute]
  end
end
```

De esta manera si creamos un objeto cualquiera y le intentamos aplicar:

```
helado = Helado.new
helado.color = azul
```

Se lanzará method_missing y se ejecutará esa porción de código. En este caso se crearía un atributo nuevo llamado *color* al que se asignaría el valor *azul*.

3.1.3. Una sintaxis muy especial

Si empezamos a programar en Ruby por primera vez, lo primero que nos llama la atención es su sintaxis. En un principio fue diseñado para ofrecer una sintaxis natural, similar al lenguaje humano (inglés) pero no simple.

A pesar de esto, el tratamiento de ciertas operaciones o tratamiendo de estructuras de datos como símbolos resulta en un principio complicada y necesita de la consulta de ejemplos o documentación.

Pasado un tiempo, la gente suele apreciar enormemente la sintaxis de Ruby, pues en muy pocas líneas se pueden hacer muchas cosas, y de manera muy natural.

Si observamos algunos ejemplos, podemos ver cómo se parece el lenguaje al inglés.

```
cadena = "hola Mundo"
cadena.upcase

#OUTPUT> HOLA MUNDO
```

También en el tratamiento de estructuras de control:

```
estanteria = ["Harry Potter", "Asterix"]
estanteria.each do |libro|
    puts ("Tengo el libro titulado " + libro)
end

#OUTPUT> Tengo el libro titulado Harry Potter
#OUTPUT> Tengo el libro titulado Asterix
```

En inglés lo reconoceríamos como algo natural, para cada elemento de este conjunto hago algo.

3.1.4. Distribuido en gemas

No podemos hablar de Ruby sin dedicarle un apartado a la peculiar manera que tiene de distribuir sus bibliotecas o paquetes, las gemas. Estas proporcionan un formato estándar y autocontenido denominado "gem" que facilita la instalación y distribución de estos vía internet.

Podemos instalarlas mediante el comando "gem install xxxxx" el cual buscará la ultima versión disponible del mismo y tratará de instalarla.

3.1.5. Otras características

- Puede ser ejecutado bajo cualquier sistema operativo, con interpretes escritos en Java, C, ensamblador, Smalltalk, etc.
- Manejo de excepciones similar al que usan Java o Python. [11]
- Un gran sistema recolector de objetos no referenciados (objetos basura).
- Manejo de hilos independiente del sistema operativo.
- *Duck typing*. El conjunto de propiedades y métodos es lo que realmente representa al objeto, no su clase. [13]

3.2. Ruby on Rails

Ruby on Rails (RoR), o simplemente Rails es un framework para el desarrollo de aplicaciones web escritas en Ruby. Es un framework muy potente que se apoya en el uso de patrones de ingeniería de software y paradigmas incluyendo arquitectura MVC, Active Record y convención sobre configuración.

Rails está diseñado alrededor de la idea de que existe una mejor manera de hacer las cosas y de alguna manera te "obliga" a hacerlas así [14]. Con esto también intenta evitar determinadas malas prácticas. Si uno aprende a hacer las cosas a la manera Rails, tendrá una mejor experiencia de desarrollo y ganará velocidad y calidad.

Esta idea de seguir los raíles que Rails te ofrece, de no reinventar la rueda, se ve apoyada además por la gran cantidad de módulos (gemas) que existen para Ruby on Rails y que nos ahorran tiempo, esfuerzo y añaden calidad al código.

3.2.1. Convención sobre configuración

Uno de los principios sobre los que se fundamenta Ruby on Rails es la convención sobre la configuración. Esto se refiere a una serie de reglas y nombres a las cuales nos tenemos que ceñir al trabajar con Rails [15].

Un buen ejemplo de esto serían los nombres de las tablas de base de datos y el mapeo (traducción) a clases. Rails utiliza la convención de utilizar el nombre de la clase pluralizado para nombrar a la tabla. De esta manera la clase "usuario" estária relacionada con una tabla "usuarios" y Rails lo sabe sin tener que realizar ninguna configuración.

Otro ejemplo de esto podría ser, para una clase "usuario" tenemos un controlador llamado "usuarios_controller" y las vistas relacionadas con esta lógica estarían bajo una carpeta llamada "usuarios" con los nombres "new", "index" o "edit".

Es posible que estas convenciones resulten tediosas o complicadas las primeras veces, pero con el tiempo nos ahorrarán tiempo y esfuerzo.

3.2.2. Patrón MVC

MVC o Modelo-Vista-Controllador (Model-View-Controller) es un patrón arquitectónico de Ingeniería de Software para sistemas que implementan interfaces de usuario. Divide el sistema en tres partes y separa la representación de la información de la lógica de la información presentada y recibida por el usuario en su interfaz.

- **Modelo:** Esta capa es la que representa la información y los datos de la base de datos. La mayoría de veces nos encontramos una tabla y su correspondiente modelo en la aplicación. Depende del patrón de acceso a datos empleado, se implementa de una manera u otra. Rails hace uso del Active Record al que dedicamos un apartado más adelante.

- **Vista:** La vista es la capa de presentación de nuestro sistema, es la responsable de generar y mostrar la información del modelo en diferentes formatos, ya sean páginas HTML, como información XML o JSON.

- **Controllador:** El controlador es la capa encargada de conectar el modelo y la vista, actúa de intermediario y procesa los datos que vienen del modelo para pasárselos a la vista. No

debe tener lógica de acceso a la base de datos, estas operaciones deberían ser realizadas por el modelo.

En la Figura 3 podemos ver las tres capas más la base de datos. Estas capas deben relacionarse solamente con las adyacentes, excepto la base de datos, que únicamente es el modelo la que opera con ella.

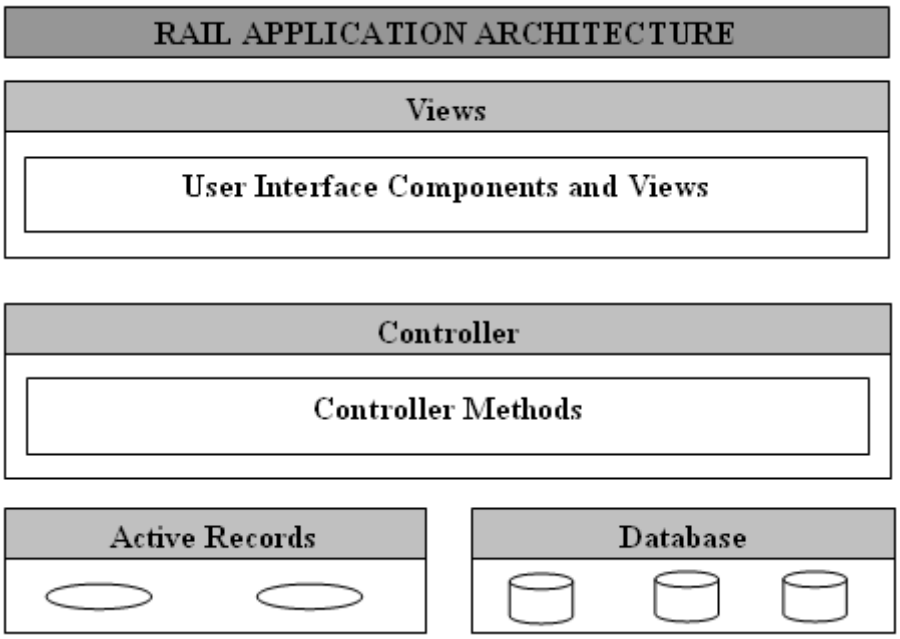

Figura 3: Arquitectura de una aplicación en Rails

Rails define una serie de buenas prácticas que harán nuestro código más mantenible, reutilizable y de calidad:

- Modelo pesado y controlador ligero.
- La lógica de negocio (datos) siempre debe estar en el modelo.
- La vista debe tener la menor cantidad de código posible.
- Haz uso de los helpers, definiciones de métodos que hacen posible su uso en la capa de la vista.
- DRY (Don't Repeat Yourself). Este principio anima a la no repetición de código. Un elemento en un sistema debería aparecer sólo una vez.

3.2.3. Active Record

Rails hace uso del patrón Active Record, un patrón de acceso a datos en el cuál es el propio modelo el que se encarga de la persistencia de los datos y del comportamiento con el que opera esos datos.

En la Figura 4 podemos apreciar cómo es el modelo el que tiene tanto los datos como la implementación de los métodos relacionados con la persistencia (CRUD).

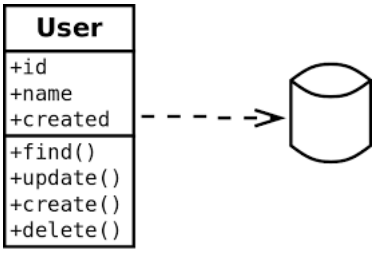

Figura 4: Patrón Active Record

La manera que tiene Ruby on Rails de implementarlo es mediante herencia. Todas las clases del modelo heredan de ActiveRecord y por tanto tienen los métodos CRUD (create, read, update, delete) de la base de datos asociada a ese modelo [16].

```
class Usuario < ActiveRecord::Base
end
```

Un ejemplo de métodos CRUD sería:

```
usuario = Usuario.create(nombre: "juan")        #CREATE
usuario = Usuario.find_by_name("juan")          #READ
usuario.update_attributes(name: "David")                    #UPDATE
usuario.destroy                                 #DELETE
```

3.2.4. Entornos de trabajo

Cuando creamos un proyecto en Ruby on Rails, este se encarga de crear tres entornos de trabajo automáticamente, con un fin muy distinto cada uno.

- **Desarrollo:** Este es el entorno por defecto, si lanzamos una instancia del servidor mediante "rails server", se montará la aplicación web con el estado que tengamos en el entorno de desarrollo. Este entorno es en el que añadimos funcionalidades y desarrollamos nuevo código.
- **Test:** Cuando vamos a pasar los test de comportamiento (de BDD), la aplicación que los pasa, por defecto los lanza en el entorno de Test. Es un entorno controlado, con una base de datos independiente de Desarrollo y que se reinicia cada vez que lanzamos los test.
- **Producción:** Este entorno es el que va a usar el usuario final, en el que desplegamos sólo si hemos desarrollado y probado en Desarrollo y hemos superado todos los test del entorno de Test. Su base de datos es independiente y no se reinicia al volver a iniciarlo.

El hecho de tener una base de datos dedicada exclusivamente a Test, nos permite interactuar con los datos de Test, de manera aislada. Los test que pasemos pueden modificar con confianza la base de datos sin afectar a otros entornos.

3.3. HTML y CSS

La aplicación web que se ha desarrollado tiene una interacción con el usuario vía navegador. La información presentada en las vistas es texto HTML formateado con hojas de estilos en cascada CSS.

3.3.1. HAML

En este proyecto se ha usado un lenguaje de abstracción sobre HTML llamado HAML (HTML Abstraction Markup Language). Este es un lenguaje de marcado ligero, con el que describir el XHTML de cualquier documento web. Uno de sus objetivos es poder tener un código elegante y por ello trata de eliminar marcas superfluas y obliga a sangrar las estructuras subyacentes. También promueve la no repetición de código vía porciones reutilizables llamados "partials" que podemos incluir en todas nuestras hojas HAML.

En Rails lo incluimos mediante la instalación de una gema. El uso de HAML en Rails está muy extendido entre su comunidad y es una de las razones por las que se ha empleado este lenguaje de marcado en el desarrollo del proyecto.

Podemos ver cómo simplifica HAML este proceso con un ejemplo:

```
<h1>Listing Books</h1>
<table>
  <tr>
    <th>Title</th>
  </tr>
<% @books.each do |book| %>
  <tr>
    <td><%= book.title %></td>
  </tr>
<% end %>
</table>
```

```
%h1 Listing Books
%table
  %tr
    %th Title
  - @books.each do |book|
    %tr
      %td= book.title
```

A la izquierda tenemos el código HTML (en rails se le pone .erb para compilar las secciones de código Ruby incrustadas) y a la derecha su equivalente en HAML.

3.4. Bases de datos

Como ya hemos explicado Rails utiliza bases de datos separadas para cada entorno y es posible configurar qué sistema de gestión de base de datos (SGBD) vamos a usar en cada uno. No todos los SGBD son iguales, algunos más pesados, otros más rápidas, por lo que debemos decidir cuál emplear en los distintos entornos. En este proyecto, se ha trabajado con bases de datos de tipo relacional.

3.4.1. Migraciones

La manera que tiene Rails de generar bases de datos y replicarlas en otros equipos o entornos es mediante las migraciones. Usan un lenguaje de dominio basado en Ruby haciendo que el esquema y los cambios sean independientes de la base de datos.

Cada migración corresponde a una versión de la base de datos, en cada una de ellas vamos implementando algún cambio, añadiendo, borrando o modificando tablas existentes. Podemos entenderlo como un control de versiones de la base de datos. Esto simplifica el traslado de la base de datos a cualquier otro entorno o equipo, de manera que si lanzamos el comando rake "db:migrate" se aplicarán todos los cambios (migraciones) en orden y al final tendremos el mismo esquema que la base datos original.

Aquí tenemos un ejemplo de una migración con la que creamos una nueva tabla

```
class Create_Usuarios < ActiveRecord::Migration
  def self.up
    create_table :usuarios do |t|
      t.string ‘‘nombre’’
    end
  end
```

En este otro ejemplo vemos una migración con la que aplicar un cambio en el tipo de una columna:

```
class Change_PrecioString < ActiveRecord::Migration
  def up
    change_table :productos do |t|
      t.change :precio, :string
    end
  end
```

3.4.2. SQLite3

Se trata de un sistema gestor de base de datos relacional basado en el estándar SQL que no garantiza integridad de dominio. Se emplea comúnmente para bases de datos locales o en el lado del cliente (navegadores, móviles).

En el desarrollo del proyecto se ha optado por utilizar este sistema de gestión de base de datos en los entornos de Desarrollo y Test por su facilidad de configuración, su portabilidad y el hecho de que es un sistema ligero.

3.4.3. PostgreSQL

PostgreSQL es un sistema mucho más pesado, con capacidad de escalado mucho mayor por lo que no es recomendable para entornos de desarrollo donde la cantidad de datos a almacenar es muy pequeña.

En el proyecto hemos decidido utilizarlo en el entorno de Producción pues aparte de su capacidad de escalabilidad, el servidor donde lo hemos alojado ofrece un buen soporte para este sistema y funciona perfectamente.

3.4.4. MySQL

La otra opción valorada para el servidor de despliegue de producción fue MySQL. Es un sistema de gestión de base de datos relacional usado ampliamente en gran cantidad de aplicaciones web. Se optó por usar PostreSQL debido a el soporte que ofrece el servidor elegido para este SGBD.

3.5. Capybara y Cucumber

En el capítulo del Marco Teórico hablábamos de TDD y de cómo mediante un lenguaje común entendible por usuario y desarrollador podíamos especificar requisitos y ganar entendimiento. En Rails este proceso se realiza mediante dos herramientas distribuidas en gemas, Capybara y Cucumber, que se necesitan la una a la otra.

3.5.1. Capybara

Capybara es una biblioteca de testing automatizado escrita en Ruby que simula la interacción de un usuario con una aplicación web. Capybara proporciona un lenguaje de dominio (DSL) basado en Ruby [17] con el que se especifican acciones cómo:

```
def sign_up_with(email, password)
  visit sign_up_path
  fill_in 'Email', with: email
  fill_in 'Password', with: password
  click_button 'Sign up'
end
```

En la cuál estamos especificando que se rellene formulario de acceso a la aplicación con email y password. Capybara es capaz de interacturar con diversos drivers tales como Selenium, Webkit o el driver por defecto de rails.

Conseguimos una herramienta de testing automático que nos permite probar la interacción con nuestra web, pero seguimos sin tener un lenguaje amable y común con el usuario. Es Cucumber realmente el encargado de proporcionarnos este marco de lenguaje común.

3.5.2. Cucumber

Cucumber es una herramienta software que corre los test de aceptación escritos en un lenguaje basado en el desarrollo orientado a comportamiento (BDD). Este busca definir un lenguaje de dominio (DSL) en inglés que el usuario (cliente) pueda entender y junto con el que desarrollar los requisitos de las features en forma de test. Cada uno de estos test corresponde con una historia de usuario.

Al estar escrito en inglés en el desarrollo de la práctica se han definido todos los test de comportamiento en inglés y así obtener un resultado más natural. Los test se escriben en ficheros separados, uno por feature, con extensión ".feature".Cucumber hace uso de ciertas palabras clave [18] para organizar la estructura de los test:

- **Feature:** En este apartado se describe qué pretende la feature. Con qué rol se va a realizar la acción, qué objetivo tiene y cómo se va a resolver ese objetivo.
- **Scenario:** Puede haber más de uno. Aquí se define el escenario o caso que nos ocupa, puede ser un caso exitoso, un caso excepcional en que se den condiciones determinadas o un caso que provoque un error.

- **Given/And/When/Then:** Estas palabras preceden cada paso del escenario. *Given* se pone cuando se definen unas condiciones previas al inicio del test de comportamiento. *And* y *When* para describir acciones tales como navegaciones o observaciones. *Then* se escribe en el último paso para introducir la condición final del test.

Con esta estructura somos capaces de dadas unas historias de usuario, escribir unos test de comportamiento para cada feature y cada posible situación que nos ocurra en esa historia de usuario.

Si observamos un caso de ejemplo:

```
Feature:  As a non-signed in user
          I want to view the register page
          So that I can create a new account

  Scenario: I navigate from sign-in to register (OK)
    Given I am on the BooksOnRails signin page
    When I follow "Register"
    And I should be on the BooksOnRails register page
```

En esta feature simulamos una interacción del usuario que situado en la página de sign-in (login) trata de seguir un enlace que le lleve a la página de registro de usuarios.

Si pasamos el test con cucumber "cucumber features/go_from_signin_to_register.feature" Cucumber tratará de pasar este test de aceptación y si existe algún fallo o falta código o elementos por implementar aparecerá en rojo el paso en el que ocurra el conflicto, el resto de pasos por debajo no llegarán a pasarse y aparecerán en azul, y los que han ocurrido antes en verde. Si todo ha ido bien, aparecerán todos en verde y podemos dar el test como aceptado.

3.6. Heroku

Los entornos de desarrollo y test de la aplicación que se ha desarrollado estaban alojados en un servidor montado en localhost, es decir en la propia máquina y no son accesibles desde fuera. En cambio para el entorno de producción tendríamos que ofrecer una web accesible desde cualquier dispositivo mediante una conexión a internet.

Heroku es una plataforma que ofrece un servicio de computación en la nube, donde poder alojar aplicaciones web basadas en distintos lenguajes. Inicialmente ofrecía soporte a Ruby on Rails actualmente también a Java, Node.js, Python, Clojure, PHP y Scala. Por defecto es gratuito pero si necesitamos ampliar capacidad tanto de almacenamiento como de cómputo o añadirle funcionalidades extra, debemos desembolsar algo de dinero.

Es una buena manera de alojar una aplicación SaaS (Software as a Service) donde la computación se realice en el servidor y el dispositivo sea solamente una interfaz donde se muestra información.

En la Figura 5 vemos cómo el usuario interactúa via consola, o mediante la administración de addons, y es Heroku el que se encarga de soportar el servicio y solicitar información.

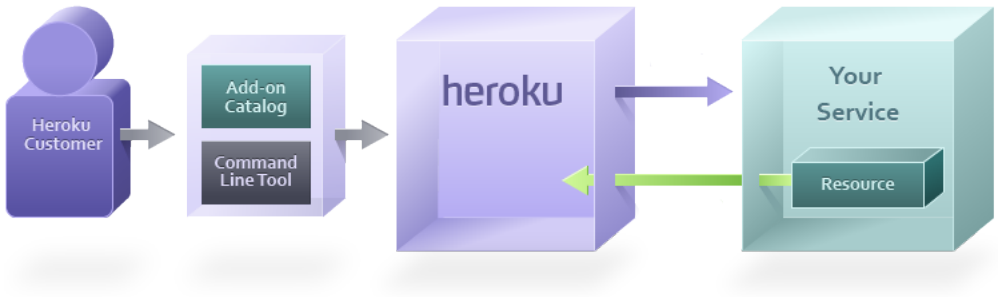

Figura 5: Relaciones entre los diferentes elementos de la plataforma

En el libro Engineering Software as a Service: An Agile Approach Using Cloud Computing [1] se dedica un apartado (Apéndice A.7) donde explica por encima cómo se realiza el despliegue de una aplicación Rails en Heroku.

3.6.1. Despliegue

El despliegue por defecto se realiza desde git. Heroku te ofrece un paquete llamado "Heroku Toolbelt" que permite mediante una serie de comandos gestionar la aplicación desde una terminal. El proceso de despliegue de una aplicación necesita de estas acciones:

1. Login en el servicio mediante:

```
$ heroku login
```

2. Clonar el repositorio de Git

```
$ heroku git:clone -a booksonrails
$ cd booksonrails
```

3. El despliegue de los cambios:

```
$ git push heroku master
```

3.6.2. Gestión de la aplicación

Heroku tiene una interfaz web donde poder observar el estado de nuestra aplicación y ver los cambios e incidencias que han ocurrido durante el tiempo.

Además ofrece mediante unos Add-ons o extensiones, ciertas funcionalidades añadidas [19], algunas de pago y otras gratuitas. Estas incluyen:

- Soporte para ciertas bases de datos. Posibilidad de guardar copias de respaldo (backups) de los datos almacenados
- Gestión y monitorización de errores y excepciones.
- Almacenamiento en caché de datos accedidos con regularidad. Mayor número de nucleos computacionales.
- Monitorización del estado, métricas y análisis de datos.
- Alertas y notificaciones.
- Herramientas de testing.
- Servicios relacionados con las redes. Seguridad de datos
- Herramientas de indexado y búsqueda.
- Email, mensajería, pagos online, procesamiento de imágenes y video.

En el caso de este proyecto, se ha tenido que hacer uso de un Add-on que ofreciera soporte para PostgreSQL. Este ofrece de manera gratuita tablas de hasta 10000 filas y 20 conexiones simultáneas a la base de datos.

3.7. Pivotal Tracker

En el capítulo dedicado a metodología ágil explicábamos los conceptos de historia de usuario, feature, sprint y la manera en la que se gestiona un proyecto de manera ágil. Pivotal Tracker es un software web que permite llevar la gestión de un proyecto basándose en todos estos elementos y trabajando de una manera determinada.

En esta aplicación se nos muestra una interfaz con opciones y diferentes apartados separados por columnas.

Estas columnas corresponde a:

- **Icebox:** Aquí se muestran las historias que tenemos planteadas pero no vamos a desarrollar a corto plazo, en el periodo de desarrollo (Sprint) actual ni siguiente.
- **Backlog:** Se muestran las historias del Sprint siguiente, ordenadas por prioridad.
- **Current:** En esta columna se muestran las historias que estamos desarrollando en el sprint actual, ordenadas en orden de prioridad.
- **Done:** En este marco se muestran las historias implementadas desde el principio del proyecto.
- **My Work:** Al trabajar en grupo, en esta pestaña podemos ver las historias asignadas (terminadas o no) a uno mismo. En el caso de este proyecto, al ser yo el único desarrollador, no tiene especial interés.
- **Epics:** Macro-historias que engloban un número de historias de manera que ofrezca una visión más amplia al equipo.

3.7.1. Concepto de velocidad

Pivotal Tracker puede ser utilizado para más cosas que ver las historias y gestionar un proyecto ágil. Además podemos medir la velocidad a la que el equipo completa historias.

Cada historia tiene asignados ciertos puntos en función de su complejidad (de 1 a 3) . Pivotal tracker expresa la velocidad de desarrollo en puntos completados por iteración [20]. Se coge la cantidad media de puntos completados en las últimas iteraciones para predecir de alguna manera el futuro de las siguientes iteraciones.

Esta información es muy útil a la hora de predecir qué historias implementar en la siguiente iteración. El equipo tiene una manera realista de poder estimar cuándo se alcanzarán ciertos hitos o milestones. Con esta información también evitamos sobreestimar o subestimar nuestras capacidades de desarrollo.

El objetivo es llegar a una velocidad constante en las últimas iteraciones con un backlog de historias ordenadas por prioridad y alcanzar un estado de predictibilidad y tranquilidad codiciado en el mundo del software.

3.7.2. Añadiendo una nueva historia

Pivotal Tracker gestiona las historias de una manera especial. Tenemos el tipo de historia, los puntos de complejidad (ver 3.7.1, concepto de velocidad), requester o solicitante, owners o

propietarios, followers a los que se notifica los cambios en la historia. Una descripción de la historia, tasks, etc.

Los tipos de historia que nos ofrece Pivotal Tracker son:

- **Feature:** Tipo de historia más común. En esta desarrollamos una funcionalidad nueva por medio de un esquema "As a - I want to - So that" al que podemos añadir los test de comportamiento.
- **Bug:** Historia orientada a solucionar un bug de una funcionalidad existente.
- **Chore:** Los chores se pueden entender como tareas más manuales, como puede ser sacar capturas, cambiar una etiqueta en una vista o modificar un tipo en un campo de base de datos.
- **Release:** Lo utilizamos cuando la tarea es desplegar cierta versión o funcionalidad en producción.

En este proyecto las historias de feature, se han introducido de la siguiente manera: En el apartado de descripción se ha escrito el enunciado de la historia, y en el apartado de tasks, se ha creado uno por escenario presente en esa feature, escribiendo todos los pasos del test de comportamiento de esa historia de usuario.

4 Desarrollo del proyecto

En este capítulo se explica el proceso del desarrollo del proyecto, desde un estudio inicial, hasta los diferentes estados y planificaciones elaboradas durante el transcurso del proyecto.

4.1. Planteamiento y estudio del problema

En este apartado explicamos por qué se ha decidido abordar este proyecto, tanto a nivel tecnológico como a nivel de información de negocio o temática.

4.1.1. Elección de la tecnología y metodología

Para el desarrollo de este proyecto, se decidió utilizar el framework Ruby on Rails, junto con una metodología ágil pues son dos elementos no demasiado comunes y entre ellos se relacionan de manera satisfactoria.

El proyecto sería por tanto una aplicación web SaaS en Ruby on Rails desarrollada con metologías ágiles.

4.1.2. Elección del problema

Una vez tuvimos elegida la tecnología y la metodología lo siguiente fue pensar qué problema abordar.

La primera idea fue algo relacionado con el tema educacional, comentamos desarrollar un juego con fines educativos. El proyecto sería gamificación (uso de juegos para actividades no recreativas) orientado a la educación. Pero dado que la premisa era desarrollar algo en Ruby on Rails y quizás Rails no sea la tecnología más adecuada para este caso, decidimos pivotar la temática.

La segunda idea fue una plataforma (también con fines educacionales) en la que poder ofrecer libros en pdf. Contaría con un visor online que tuviera fecha de caducidad, de manera que a partir de cierta fecha no pudieras seguir viendo el pdf impidiendo la posibilidad de haberlo almacenado. Esta idea requería una complejidad extra y se prefirió simplificar el proyecto.

Pensamos en hacer un sistema de compartición de libros electrónicos pero surgió el problema del copyright.

Al final se decidió desarrollar una web de gestión de reuniones para compartir libros físicos, por ejemplo orientados a clubes de lectura. De esta manera evitabamos el problema del copyright de los libros electrónicos (el préstamo o intercambio de libros físicos es legal) y establecíamos unos objetivos realistas con una complejidad abarcable.

4.1.3. Planteamiento inicial

Al estar desarrollado con métodos ágiles, el proceso no tendría los pasos típicos de *Análisis - Diseño - Implementación - Pruebas*. En cambio está separado por iteraciones o sprints en los que se van implementando diferentes historias de usuario.

Por eso lo que se decidió fue en diferentes estados del proyecto, tomar una captura general del estado del proyecto, capturas que reflejen cambios en la interfaz, las historias de usuario desarrolladas, los test de comportamiento realizados, etc. Resultando en un total de 4 iteraciones

Se planteó tener reuniones regulares cada 2-3 semanas con carácter de supervisión y consulta en caso de existencia de dudas o problemas.

4.1.4. Cálculo inicial del presupuesto

Como en todo proyecto, es necesario una primera estimación del coste de desarrollo antes de llevarlo a cabo. Al final del documento de seguimiento del desarrollo del proyecto software, se estimará el coste final del proyecto de manera que se pueda valorar si este se desvía sobre lo presupuestado o no, y valorar este resultado.

Este proyecto está englobado como una asignatura de 12 créditos ECTS. Dado que un crédito ECTS equivale a 25 horas de trabajo del alumno, los 12 créditos de la asignatura equivalen a 300 horas de trabajo personal.

Suponemos entonces que en el desarrollo del proyecto vamos a emplear 300 horas de trabajo. Para calcular el coste debemos saber el precio de la hora de trabajo de una persona.

Para calcular el precio de las hora de trabajo, podemos fijarnos en el sueldo medio de unas prácticas de empresa remuneradas, alrededor de 5 euros/hora. Dado que el desarrollador aún no es un profesional titulado, tomar ese coste de hora de trabajo sería adecuado.

Por tanto el coste a priori del desarollo del proyecto software sería:

```
Número de horas: 300 horas
Remuneración del desarrollador: 5 euros/hora

300 horas * 5 euros/hora = 1500 euros
```

En el apartado "Cálculo final del presupuesto" (ver 4.7) estimamos el coste real del desarrollo de la aplicación y lo valoramos sobre el calculado aquí.

4.2. Iteración 0

La iteración 0 es la iteración inicial, es una iteración particular, pues a diferencia del resto, en esta no tenemos una aplicación creada inicialmente. La configuración de equipos y entornos también se abarcará en esta iteración

Tenemos que reunirnos con el cliente para establecer las primeras historias. También tendremos que realizar unos bocetos o prototipos visuales que de alguna manera permitan al cliente interactuar con ellos para darle una futura estructura a la aplicación.

4.2.1. Configuración del entorno de trabajo

Como hemos dicho, en esta iteración empezamos sin un equipo configurado para el desarrollo de este tipo de proyectos.

4.2.1.1. Equipo

Los equipos que han participado en el desarrollo de la aplicación y alojamiento de entornos de desarrollo y test son

- Un ordenador con Windows 8.1 (64 bit). Procesador i5 3570K @3.4GHz, 8GB RAM DDR3, tarjeta gráfica AMD HD 7870, 500GB HDD. Este Equipo es el encargado de alojar una VM (máquina virtual).
- Una VM de Oracle VirtualBox con Ubuntu 12.04(LTS)(32 bit) con 1 núcleo @3.4GHz, 24MB de memoria de vídeo, 9GB de disco y 3GB RAM. En este equipo se ha realizado todo el desarrollo, tests y ha soportado los servidores de desarrollo y test en localhost.
- Un ipad 2 como dispositivo de interacción con el cliente y con la supervisora del proyecto en las reuniones.
- Pantalla 23" con resolución 1920 x 1080 donde se ha desarrollado y probado la aplicación web.

4.2.1.2. Software, herramientas y versiones

Hemos tenido que instalar también todo el software y herramientas necesarias para desarrollar el proyecto, teniendo en cuenta las posibles incompatibilidades entre versiones que pudieran surgir.

El software que hemos instalado, y su versión:

- Ruby 1.9.3
- Rails 3.2.16
- Cucumber 1.3.17
- Capybara 2.4.4
- HAML 4.0.4
- SQLite3 1.3.10
- PostgreSQL 0.17.1

También documentamos las herramientas que hemos utilizado y si es posible, su versión:

- Sublime Text 2.0.2
- Firefox 32.0
- Pivotal Tracker
- Balsamiq
- Heroku
- Github
- Dropbox

Algunas de las herramientas no han necesitado de instalación si no que se encuentran en un servidor y se ha accedido a ellas vía navegador.

4.2.2. Funcionalidades generales

En la primera reunión con el cliente (yo tomo el papel de cliente en el desarrollo de este proyecto) se analiza el problema y se extraen una serie de funcionalidades y conceptos que ha de cubrir nuestra aplicación.

Al ser un sistema de gestión de reuniones, en las que decimos qué libros vamos a llevar, deberemos almacenar una colección de ejemplares para cada usuario. También deberá existir un catálogo general libros. En el apartado de las reuniones, se podrán gestionar creándolas, viéndolas o apuntándose a ellas.

Se ha decidido que la web tenga un sistema de gestión del catálogo colaborativo, de manera que sea todo el mundo el que pueda añadir libros o editarlos. Esto podría ser controlado en algún momento mediante administradores.

Esta información nos servirá para crear unos mockups o prototipos visuales.

4.2.3. Prototipado visual

En base a la información de conceptos y funcionalidades generales extraídos en la primera reunión, se crean unos prototipos visuales interactivos, con los que el usuario puede realizar la navegación que ofrecerá la página sin el look and feel final y sin funcionalidades implementadas.

Estos prototipos tienen varios fines:

- Ayudarnos a generar nuevas historias de usuario.
- Servirnos como guías de diseño para la estructura de la interfaz.
- Mostrar al cliente un borrador susceptible a cambios, mejoras o críticas constructivas.

En el primer prototipo visual introducimos el elemento del menubar superior. Este nos ayuda a tener rápido acceso a las funciones principales de la apliación, a la vez que nos sitúa (mediante una barra de color debajo del apartado) en la sección en que nos encontramos.

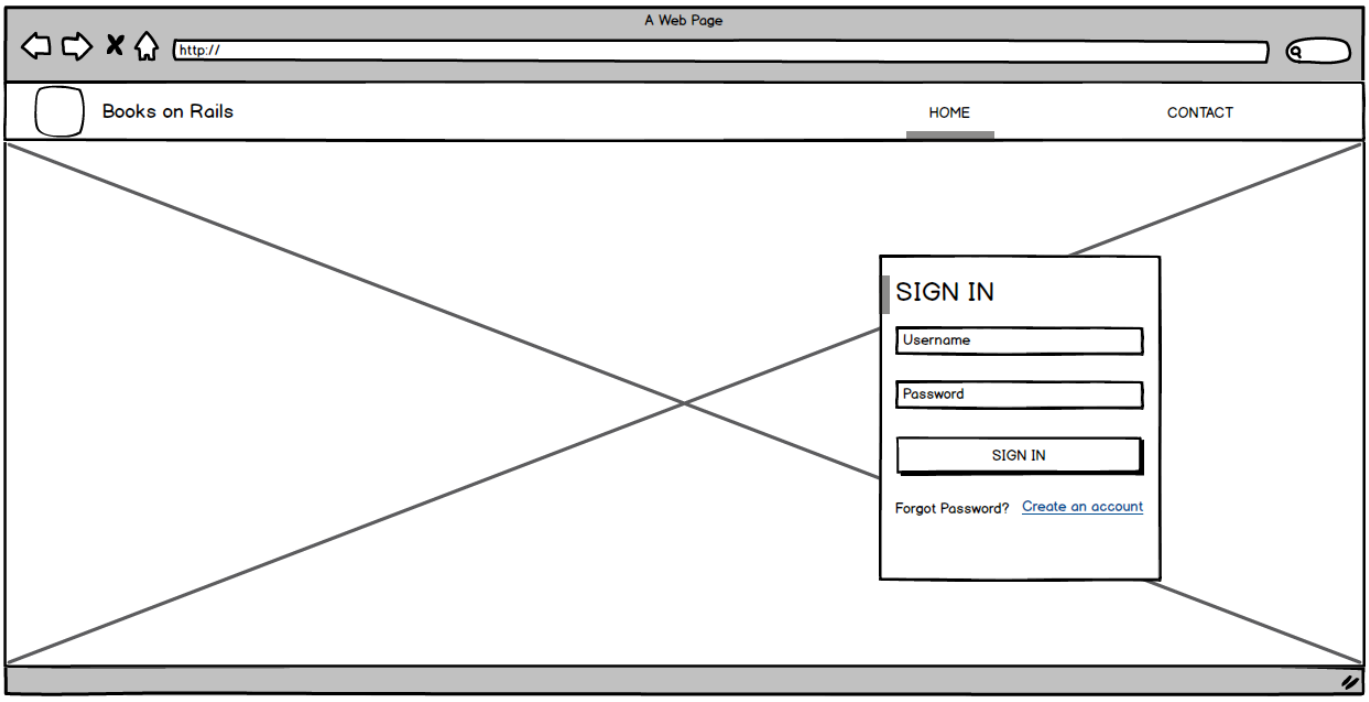

Figura 6: Página de ingreso

Esta es la página que nos encontramos al acceder a la aplicación. Se trata de un ingreso mediante la introducción de usuario y contraseña (ver Figura 6). Necesitaremos para ello una cuenta que se crea siguiendo el link de "Create an account".

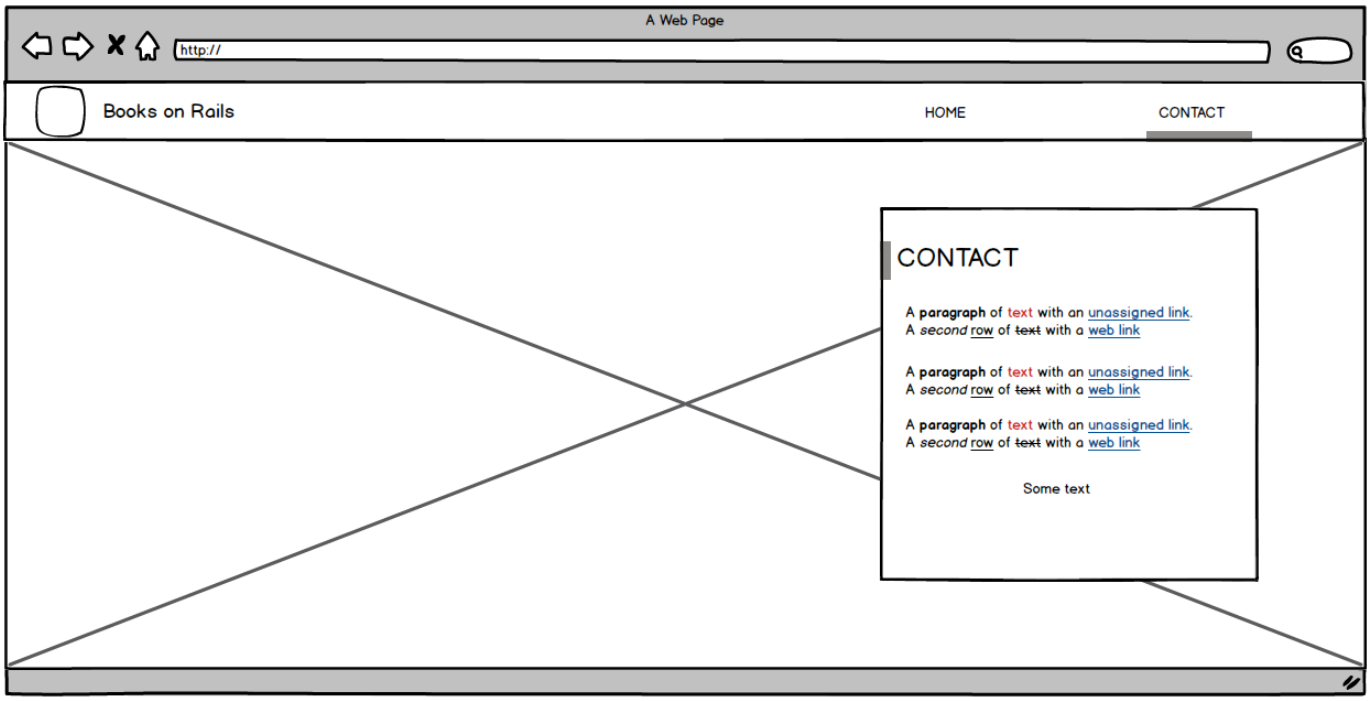

Figura 7: Página de Contacto

En esta página (ver Figura 7) tenemos la información de contacto, accesible desde la página de ingreso. Es importante en caso de que un usuario desease contactar con el desarrollador.

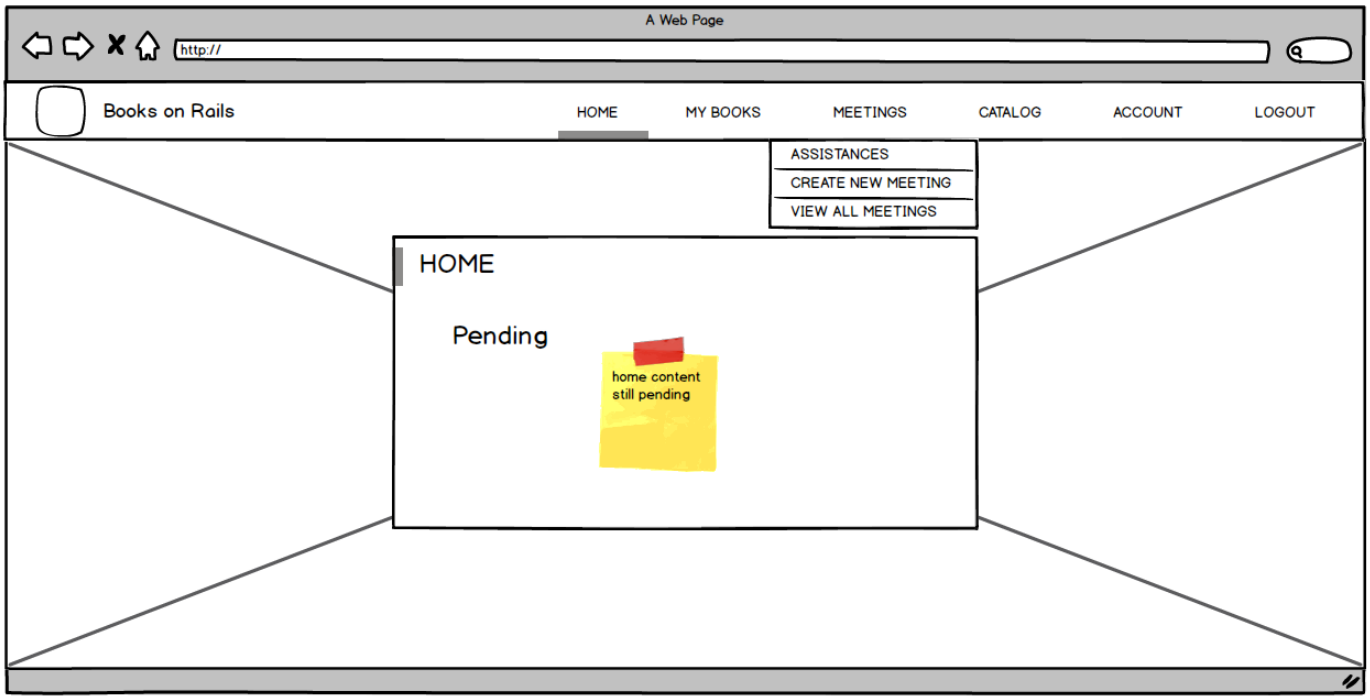

Figura 8: Página de bienvenida o Home.

En esta página (ver Figura 8) vemos la página de bienvenida, a la que accede el usuario tras ingresar en el sistema. Al ser prototipos en un estado temprano, no se tenía muy claro qué clase de información incluir en esta página. Podemos ver que el menú superior ha cambiado, introduciendo las opciones principales de la aplicación.

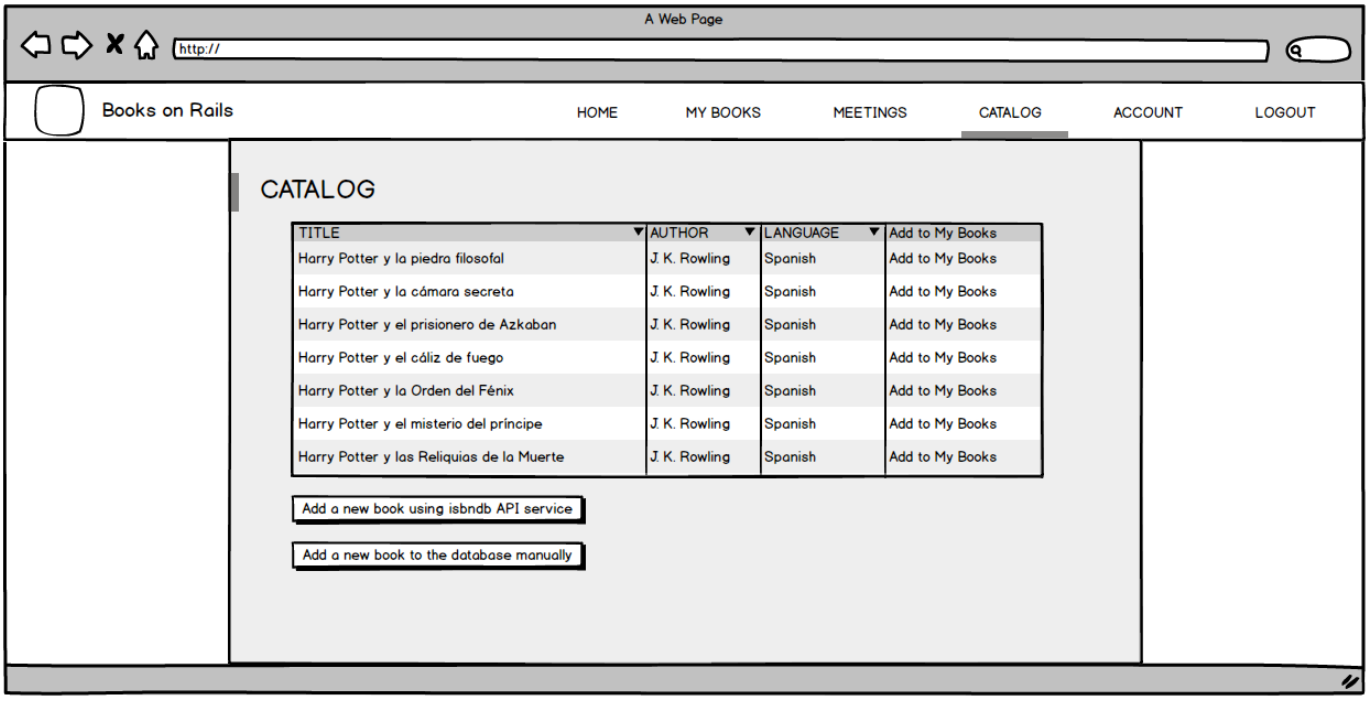

Figura 9: Catálogo de libros

En esta página (ver Figura 9) mostramos la lista de libros existentes en el catálogo de la aplicación. Si se pulsa “Add to My Books” se añadirá un ejemplar de este libro en tu colección particular. Además se ofrecen opciones para añadir libros al catálogo.

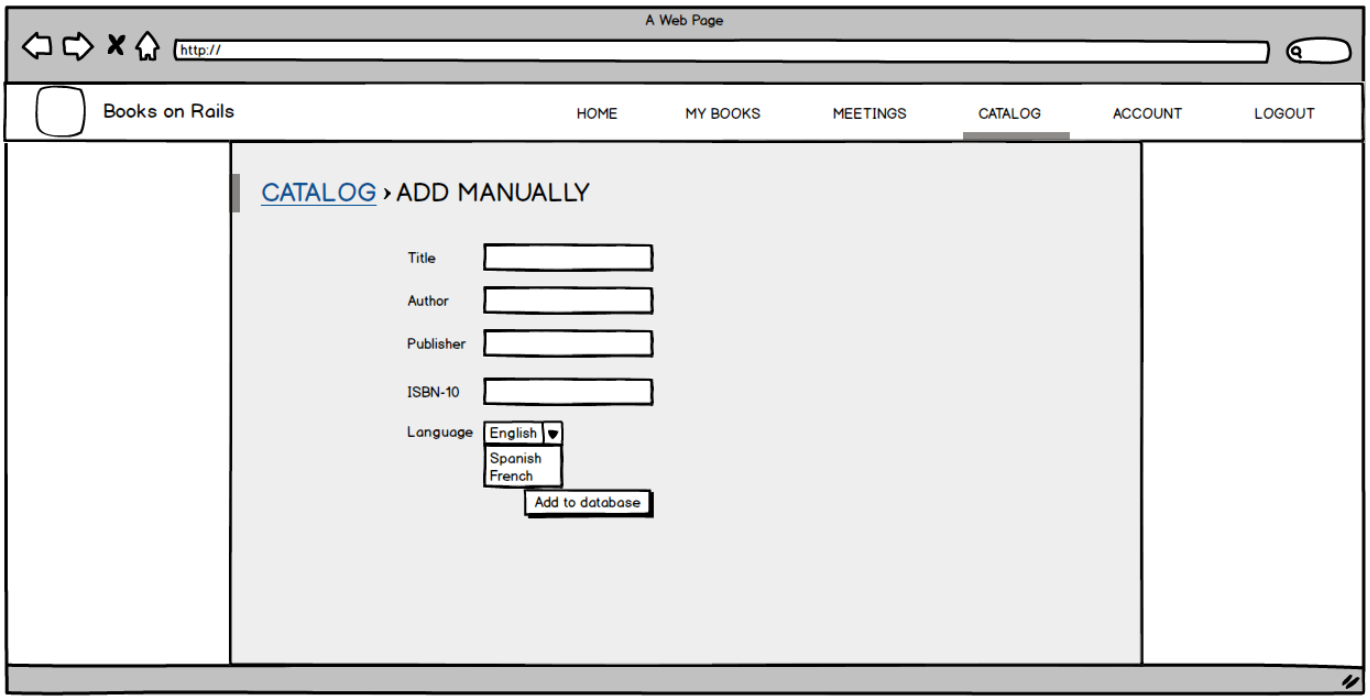

Figura 10: Añadir un libro a mano

Aquí (ver Figura 10) podemos añadir un libro introduciendo manualmente los datos en un formulario. Este proceso está disponible para todo el mundo al tratarse de una web colaborativa.

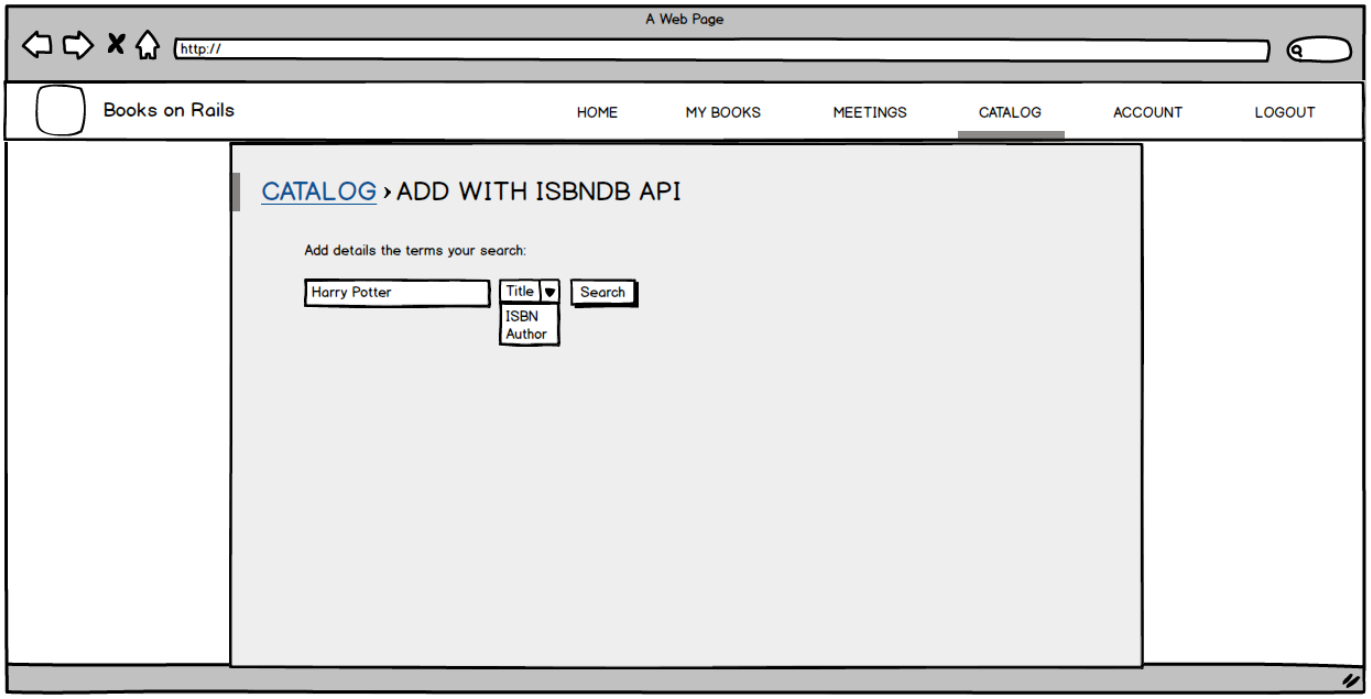

Figura 11: Añadir un libro via API de isbndb

Si escogemos la opción de añadir un libro con la API de isbndb, podemos añadir libros sin tener que rellenar sus datos.

Utilizamos una base de datos ya existente (ISBNDB) para indexar todos los libros correspondientes con una búsqueda (ver Figura 11).

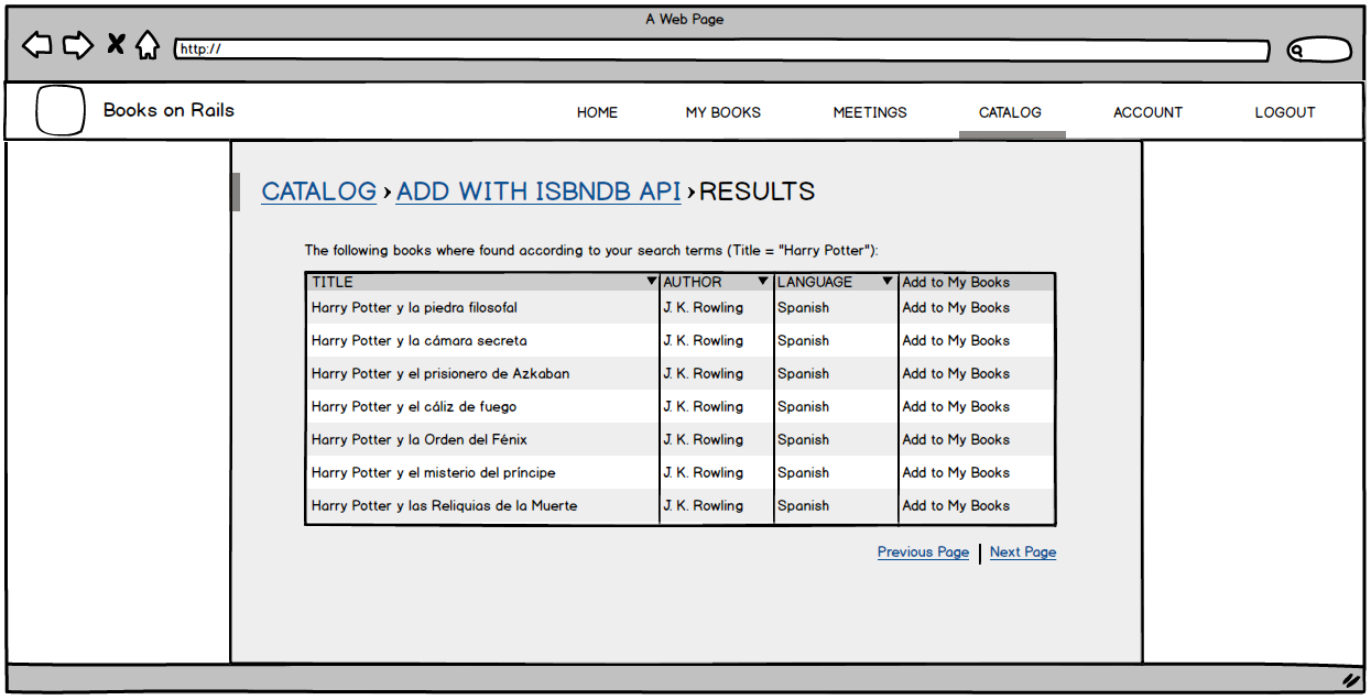

Figura 12: Resultados de la API de isbndb

Aquí (ver Figura 12) se muestran todos los resultados que concuerden con los criterios de búsqueda (vía API de isbndb). A esta página se conduce al pulsar "Search" en la página de la Figura 11.

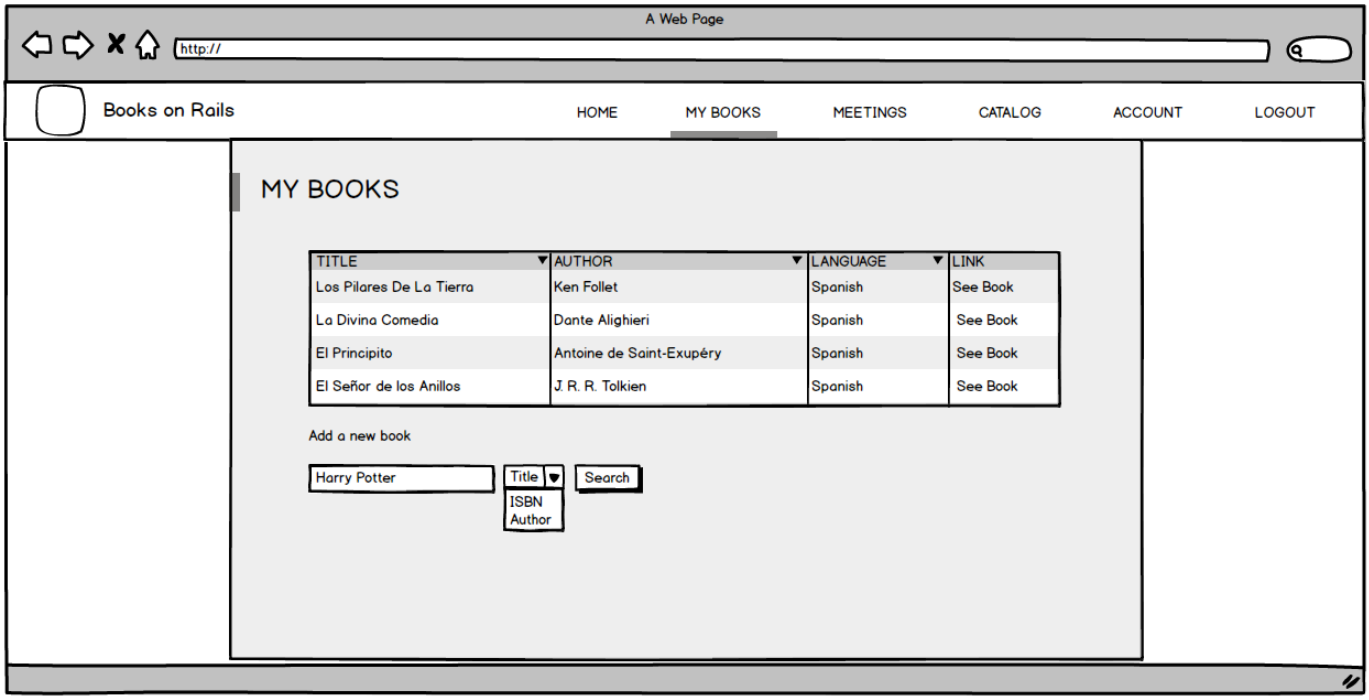

Figura 13: Los ejemplares de mi colección

En esta página (ver Figura 13) vemos la lista de ejemplares que tenemos en nuestra colección particular. Se muestran los libros de Catalog que hemos marcado con "Add to My Books". Además se ofrece una opción de búsqueda para añadir ejemplares a la colección rápidamente (de libros que existan previamente en el catálogo general)

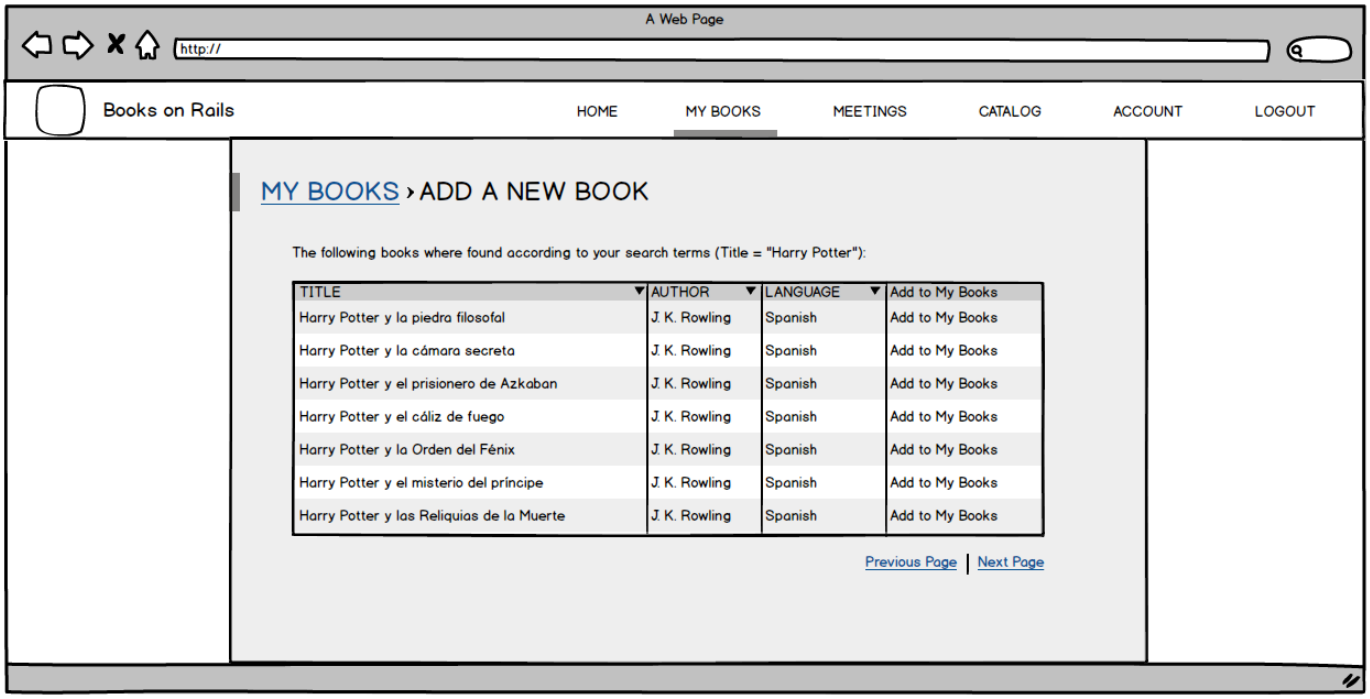

Figura 14: Búsqueda de libros desde mi colección

Si hemos decidido buscar un libro desde My Books, se mostrará esta página (ver Figura 14) en la que muestra una lista con los libros acordes con la búsqueda. Una vez más podemos añadirlos a nuestra colección haciendo click en “Add to My Books” en la fila correspondiente a cada uno de ellos.

Figura 15: Página de meetings

Aquí (ver Figura 15) mostramos las opciones relativas a la gestión de meetings en forma de botones.

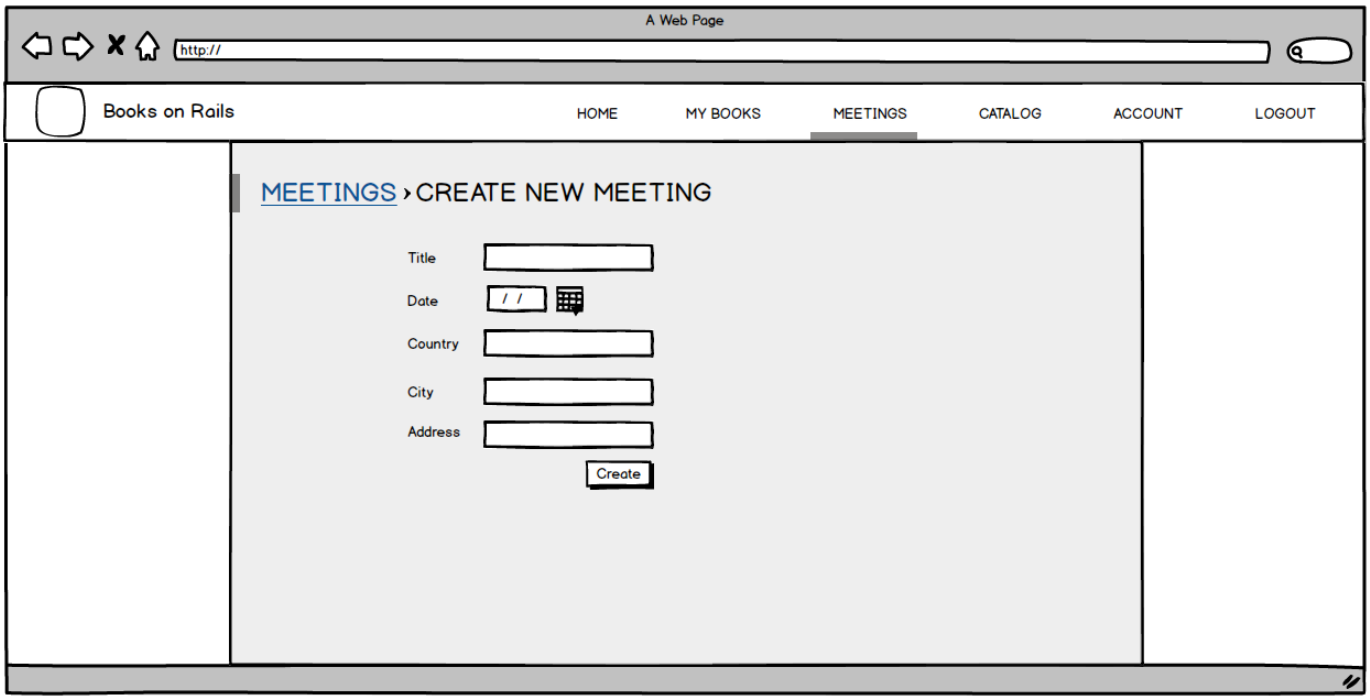

Figura 16: Crear un nuevo meeting

Podemos crear un nuevo meeting si en la página de la Figura 15 hacemos click en "Create new meeting". Rellenamos los datos en un formulario para crear un meeting de forma manual (ver Figura 16).

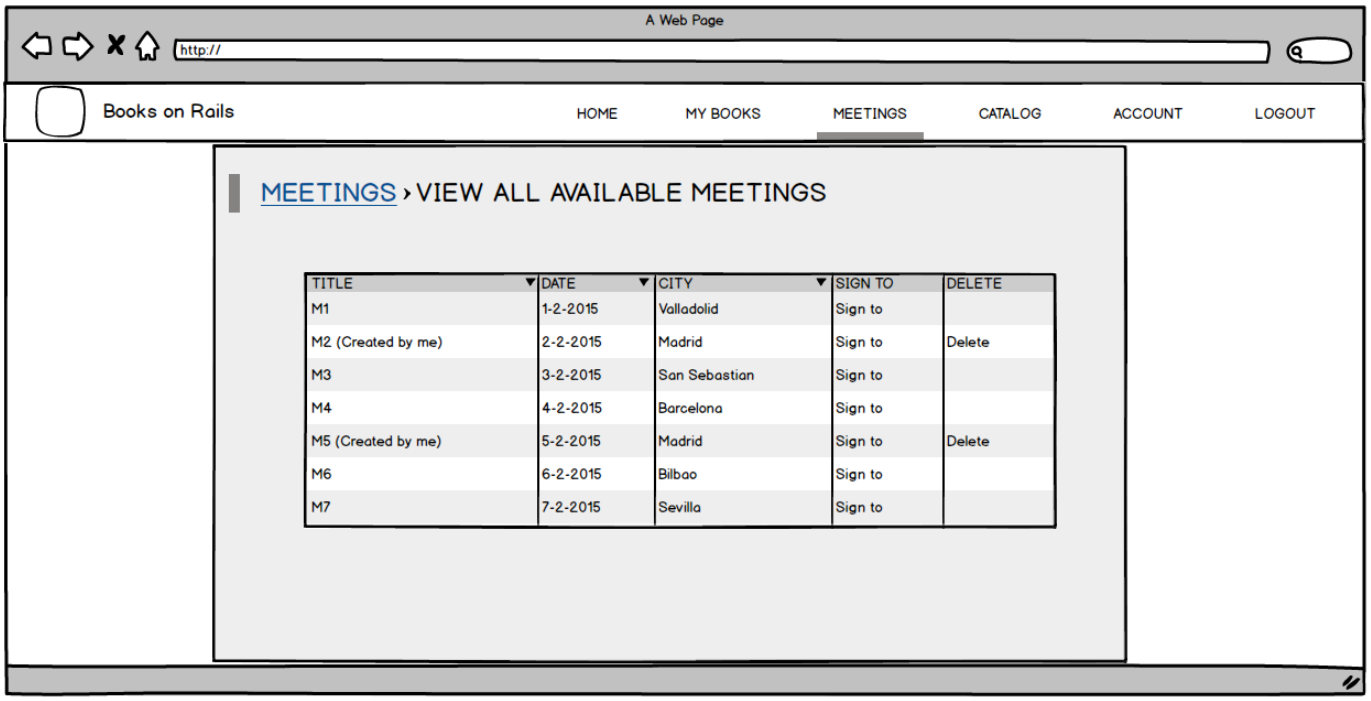

Figura 17: Lista de meetings disponibles

Si hemos hecho click en "View all meetings" en la página de la Figura 15, veremos la lista de todos los meetings disponibles en la aplicación. Podemos apuntarnos a un meeting haciendo click en "Sign to".

Figura 18: Asistiendo a un meeting

Al apuntarnos a un meeting se nos muestra esta página (ver Figura 21) con opciones que incluyen establecer qué ejemplares vamos a llevar a ese meeting.

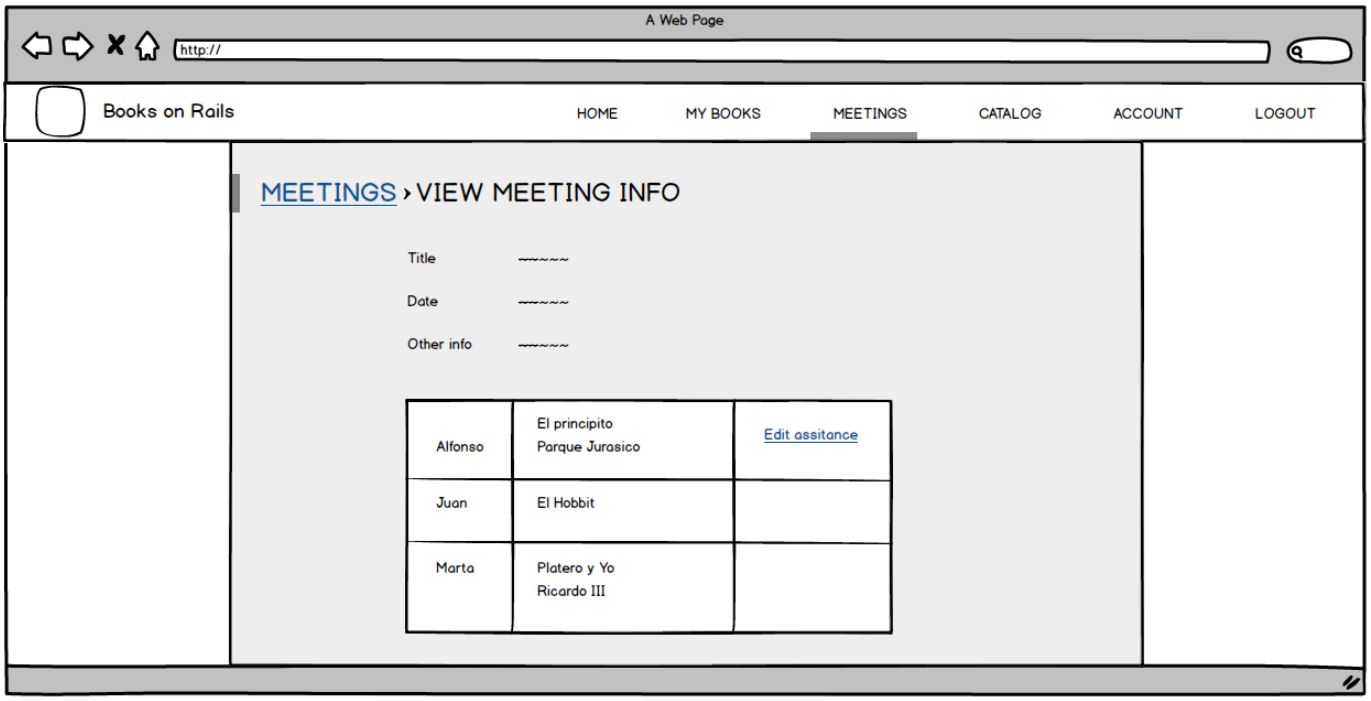

Figura 19: Información de un meeting

Si en la lista de meetings hacemos click encima del nombre de cada uno, accederemos a la información del meeting en cuestión (ver Figura 19).

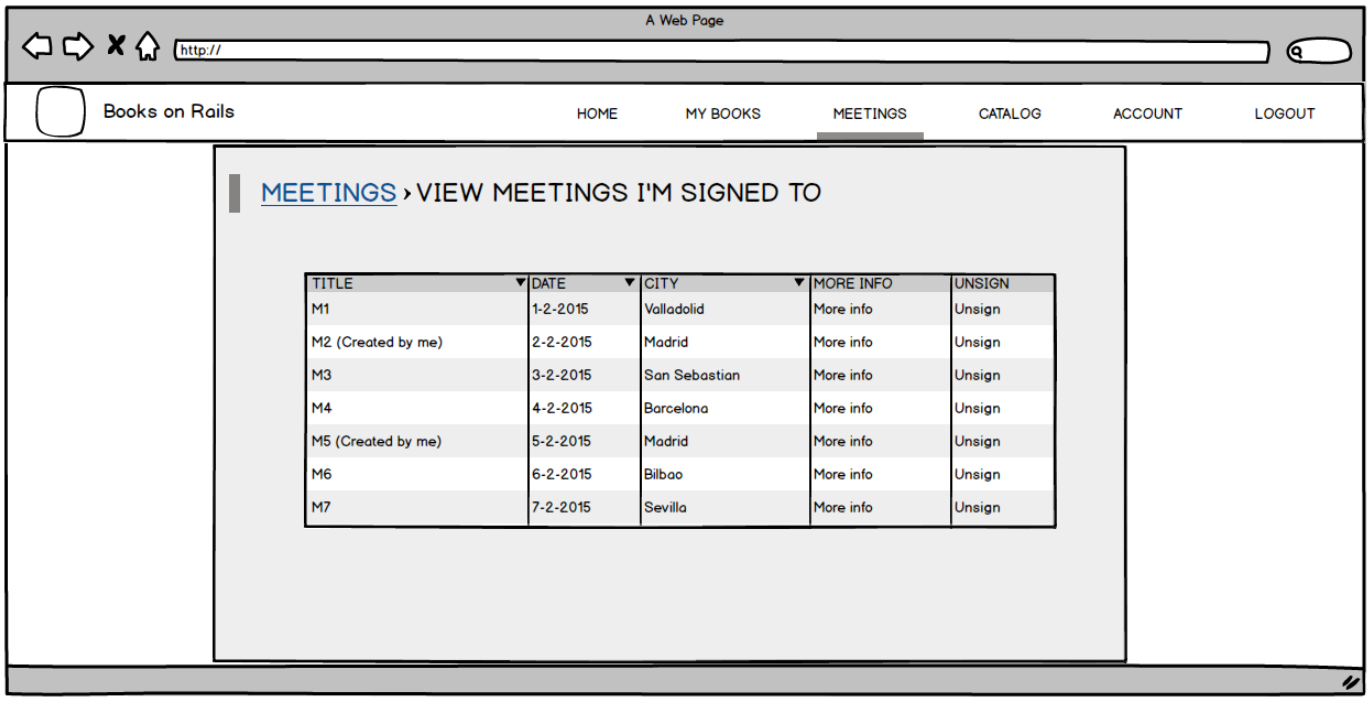

Figura 20: Lista de meetings a los que se va a asistir

Si en la lista de opciones de meetings seleccionamos “View meetings you are signed to” veremos una lista compuesta por los meeting a los que nos hemos apuntado.

4.2.4. Primeras historias de usuario

Tras una primera reunión con el cliente (yo tomo el papel de cliente en el desarrollo de este proyecto) se escriben las primeras historias de usuario que cubrirán una serie de funcionalidades iniciales.

El ingreso a la aplicación de un usuario con su cuenta:

```
Story: User sign in
        As a non signed in user in Sign In page
        So that I can see access the content of the site
        I want to sign in with an existing account
```

La creación de una nueva cuenta de usuario:

```
Story: User registration
        As a non signed in user in Sign In page
        So that I can access the application any time
        I want to navigate to create a new account
```

Añadir un nuevo libro al catálogo:

```
Story: Adding new books to catalog
        As a signed in user in Catalog page
        So that make the database collection of books bigger
        I want to add a new book
```

Editar un libro existente del catálogo

```
Story: Editing a book in catalog
        As a signed in user in Catalog page
        So that I modify the data of an existing book
        I want edit an existing book
```

Añadir un ejemplar a tu colección:

```
Story: Add exemplar to my books
        As a signed in user in home page
        So that I can have more books in my collection
        I want to add an exemplar of a book
```

Quitando un ejemplar de tu colección:

```
Story: Removing exemplar from my books
        As a signed in user on the home page
        So that I can eliminate some exemplars from my collection
        I want to remove an exemplar
```

Diferentes historias, cada una representa la navegación de una página a las demás.

```
Story: Navigation from sign in to contact
        As a non logged user in Sign In page
        So that I can see contact information
        I want to navigate to the Contact page
```

```
Story: Navigation from contact to sign in
        As a non logged user on the Contact page
        So that I can see the Sign In page
        I want navigate to the Sign In page
```

```
Story: Navigation from home to catalog
        As a signed in user in Home page
        So that I can see the Catalog page
        I want navigate to the Catalog page
```

```
Story: Navigation from home to account
        As a signed in user in Home page
        So that I can see the Account page
        I want navigate to the Account page
```

```
Story: Navigation from catalog to home
        As a signed in user in Catalog page
        So that I can see the Home page
        I want navigate to the Home page
```

```
Story: Navigation from catalog to account
        As a signed in user in Catalog page
        So that I can see the Account page
        I want navigate to the Account page
```

```
Story: Navigation from account to catalog
        As a signed in user in Account page
        So that I can see the Catalog page
        I want navigate to the Catalog page
```

```
Story: Navigation from account to home
        As a signed in user in Account page
        So that I can see the Home page
        I want navigate to the Home page
```

Las siguientes corresponden con el cierre de la sesión de tu cuenta de la aplicación.

```
Story: Logout from home
        As a signed in user in Home page
        So that I end the use of the site with my account
        I want to logout
```

```
Story: Logout from account
        As a signed in user in Account page
        So that I end the use of the site with my account
        I want to logout
```

```
Story: Logout from catalog
        As a signed in user in Catalog page
        So that I end the use of the site with my account
        I want to logout
```

Estas son la historias generadas que se implementarán hasta la iteración 1.

4.2.5. Diagrama de clases y tablas

En base a las historias planteadas y el planteamiento de funcionalidades generales, esbozamos un diagrama de clases donde se sitúan las clases presentes en la aplicación, los atributos o campos de cada clase y las relaciones y cardinalidades entre ellas.

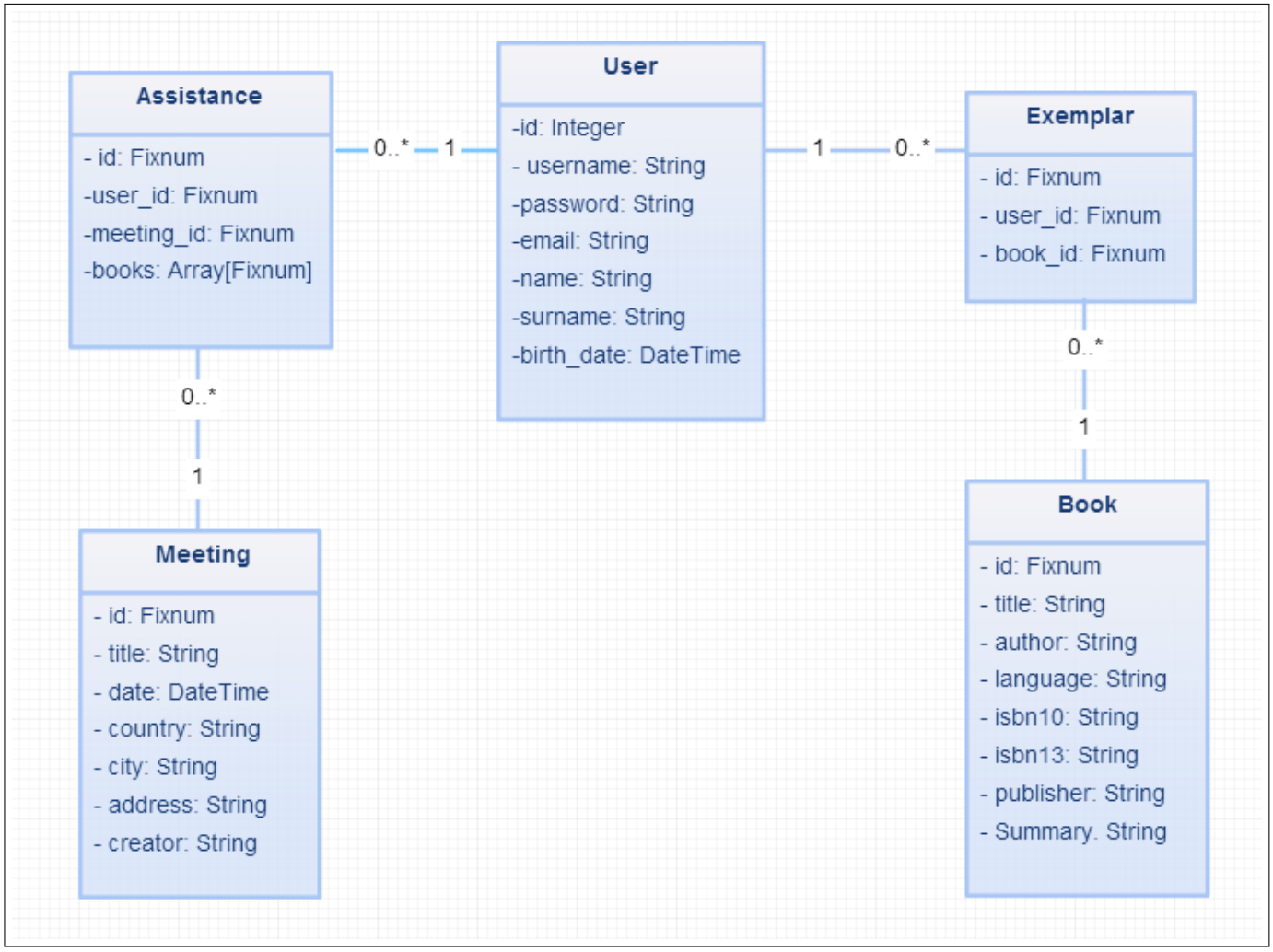

Figura 21: Diagrama de clases

Rails crea una tabla por modelo (clase del dominio) por lo que las tablas serán cada una la equivalente a su modelo y cada atributo tendrá su campo correspondiente con el tipo adecuado.

4.3. Iteración 1

Esta iteración es la primera en la que se realiza implementación y se obtiene una demo funcional que enseñar al cliente. Esto permitirá realizar cambios en etapas tempranas del desarrollo, lo que generará menos problemas.

Lo primero que se hizo fueron los test de comportamiento asociados a cada historia de usuario, a partir de los cuales añadir código para cumplirlos e implementarlos, refactorizar ese código nuevo y volverlos a pasar.

4.3.1. Test de comportamiento

Todos los test que se presentan en este apartado son test que funcionan si los lanzamos con cucumber, esto nos valida lo pactado con el usuario de una manera formal. El texto en color verde corresponde a los pasos que intenta reproducir cucumber cuando lo lanzamos con esta feature. Cada uno de ellos equivale a una acción (rellenar un campo, hacer click en un botón, seguir un link, ver cierto texto en el contenido de la web...). Los test 1 y 2 corresponde a los test que validan la navegación entre enlaces accesibles desde la página de acceso a la aplicación y la página de contacto, ambas accesibles sin tener que acceder con una cuenta.

En base a los test 1 y 2 se crearon las vistas correspondientes y el código que deberían tener para que al pasar cucumber el test, los pasos salieran verdes.

```
 Test 1
Feature: As a non logged user in Sign In page
         So that I can see contact information
         I want to navigate to the Contact page

Scenario: Go from Sign In to Contact
  Given I am on the BooksOnRails signin page
  And I should see a link titled "CONTACT" that goes to "/contact"
  And I follow "CONTACT"
  Then I should be on the BooksOnRails contact page
```

```
 Test 2
Feature: As a non logged user on the Contact page
         So that I can see the Sign In page
         I want navigate to the Sign In page

  Scenario: Go from Contact to Sign In
  Given I am on the BooksOnRails contact page
  And I should see a link titled "SIGN IN" that goes to "/sessions/new"
  And I follow "SIGN IN"
  Then I should be on the BooksOnRails signin page
```

El test 3 definimos los pasos para crear una nueva cuenta de usuario que validan el comportamiento del funcionamiento de registro en la aplicación. Podemos ver cómo los pasos son seguir un enlace, rellenar los campos correspondientes y pulsar el botón Registrar.

A partir del test 3, se creó la vista de registro de usuarios y el formulario con los campos correspondientes para cumplir el test. También el modelo y controlador de usuarios.

```
 Test 3
Feature: As a non signed in user in Sign In page
         So that I can access the application any time
         I want to navigate to create a new account

Scenario: Register a new user (successful)
  Given I am on the BooksOnRails signin page
  When I follow "Register"
  And I should be on the BooksOnRails register page
  And I fill in "First Name" with "Juan"
  And I fill in "Second Name" with "Lopez"
  And I fill in "Username" with "juanlop"
  And I fill in "Email" with "juanlop@hmail.com"
  And I fill in "Password" with "1234"
  And I press "Register"
  Then I should be on the BooksOnRails signin page

Scenario: Register a new user (error: user already exists)
  Given there is a user with username "juanlop"
  Given I am on the BooksOnRails signin page
  When I follow "Register"
  And I should be on the BooksOnRails register page
  And I fill in "First Name" with "Juan"
  And I fill in "Second Name" with "Lopez"
  And I fill in "Username" with "juanlop"
  And I fill in "Email" with "juanlop@hmail.com"
  And I fill in "Password" with "1234"
  And I press "Register"
  Then I should be on the BooksOnRails register page
  And I should see "User not created, username is already taken"
```

En el test 4 realizamos los pasos del acceso al sistema con una cuenta ya creada. A cucumber le decimos que dado que ya existe una cuenta con usuario “juanlop”, email “juanlop@hmail.com” y password “1234” como premisa, rellene los campos con esos datos y acceda a la aplicación, a la página de bienvenida (home).

Con el test 4 se tuvo que crear el modelo de sesión, el de usuario y los controladores correspondientes, así como la vista de la página home.

```
 Test 4
Feature: As a non signed in user in Sign In page
         So that I can see access the content of the site
         I want to sign in with an existing account
```

```
Scenario: Sign in (successful)
  Given I register one user "juanlop" with email "juanlop@hmail.com"
  and password "1234"
  Given I am on the BooksOnRails signin page
  When I fill in "Username" with "juanlop"
  And I fill in "Password" with "1234"
  And I press "Sign in"
  Then I should be on the BooksOnRails home page

Scenario: Sign in (error: username with password not existing)
  Given I register one user "juanlop" with email "juanlop@hmail.com"
  and password "1234"
  Given I am on the BooksOnRails signin page
  When I fill in "Username" with "juanlop2"
  And I fill in "Password" with "4321"
  And I press "Sign in"
  Then I should be on the BooksOnRails home page
  And I should see "Couldn't sign in, That combination of username and
  password does not exist"
```

Este test valida el comportamiento de la acción de añadir un nuevo libro al catálogo general. Es un test similar al de registro de nuevos usuarios, salvo que esta vez es un formulario para la creación de nuevos libros (en vez de usuario) lo que tiene que rellenar cucumber con los datos que le proporcionamos en el test. Luego comprueba que el libro se halla creado.

El test 5 tiene dos escenarios posibles, en el que al añadir un nuevo libro todo va bien y en el que falla por ejemplo al comprobar un código isbn.

Para pasar el test 5, se tuvo que crear el modelo y controlador de libros y dentro de este la lógica para añadir nuevos libros al catálogo.

```
 Test 5
Feature: As a signed in user in Catalog page
         So that make the database collection of books bigger
         I want to add a new book

Scenario: Add a New Book to the Catalog (new isbn)
  Given I am signed in as user "juanlop" with password "1234"
  Given I am on the BooksOnRails home page
  And I should see a link titled "CATALOG" that goes to "/books"
  And I follow "CATALOG"
  And I follow "Add new book"
  And I fill in "Title" with "The Odyssey"
  And I fill in "Author" with "Homer"
  And I fill in "Publisher" with "Sun Publications"
```

```
  And I select "English" from "Language"
  And I fill in "isbn10" with "1234567890"
  And I fill in "isbn13" with "1234567890123"
  And I fill in "Summary" with "Greek epic poem"
  And I press "Save Changes"
  And I should be on the BooksOnRails catalog page
  Then I should see "The Odyssey"
  Then I should see "Homer"
  Then I should see "Sun Publications"

Scenario: Add a New Book to the Catalog (existing isbn)
  Given I am signed in as user "juanlop" with password "1234"
  Given I am on the BooksOnRails home page
  Given The Odyssey is in the Catalog
  And I should see a link titled "CATALOG" that goes to "/books"
  And I follow "CATALOG"
  And I follow "Add new book"
  And I fill in "Title" with "The Odyssey"
  And I fill in "Author" with "Homer"
  And I fill in "Publisher" with "Sun Publications"
  And I select "English" from "Language"
  And I fill in "isbn10" with "1234567890"
  And I fill in "isbn13" with "1234567890123"
  And I fill in "Summary" with "Greek epic poem"
  And I press "Save Changes"
  Then I should see "Errors prevented this book from being created"
```

El test 6 trata de editar los datos de un libro existente (The Odyssey) haciendo click en "Edit info" y cambiando el dato de Publisher, luego comprueba que el campo haya sido actualizado.

Al escribir el test 6 se hizo necesaria la implementación del sistema de actualización de libros en el controlador de libros para cumplir todos los pasos. También se tuvo que crear la vista de actualización de libros.

```
 Test 6
Feature: As a signed in user in Catalog page
         So that I modify the data of an existing book
         I want edit an existing book

Scenario: Add a New Book to the Catalog
  Given there is a book with title "The Odyssey" and author "Homer"
  and publisher "Sun Publications"
  Given I am signed in as user "juanlop" with password "1234"
  Given I am on the BooksOnRails home page
  And I should see a link titled "CATALOG" that goes to "/books"
```

```
  And I follow "CATALOG"
  And I follow "The Odyssey"
  And I follow "Edit info"
  And I fill in "Publisher" with "Athens Publications"
  And I press "Save Changes"
  Then I should see "Athens Publications"
  And I follow "Back to book list"
  Then I should see "Athens Publications"
```

El test 7 se trata de añadir uno de los libros existentes en el catálogo a la colección personal del usuario como un nuevo ejemplar. Los pasos que trata de realizar son hacer click en "Add to My Books" y luego comprobar que efectivamente ese libro está visible en My Books.

El test 7 nos hizo implementar el modelo de ejemplar, la vista de todos los ejemplares del usuario, el botón de "Add to My Books" y la lógica asociada a añadir un nuevo ejemplar a la colección personal.

```
 Test 7
Feature: As a signed in user in home page
         So that I can have more books in my collection
         I want to add an exemplar of a book

Scenario: Add a non-existent Exemplar to My Books
  Given there is a book with title "The Odyssey" and author "Homer"
  and publisher "Sun Publications"
  Given I am signed in as user "juanlop" with password "1234"
  Given I am on the BooksOnRails home page
  And I should see a link titled "CATALOG" that goes to "/books"
  And I follow "CATALOG"
  And I press "Add to My Books"
  Then I should see "Book 'The Odyssey' has been added to My Books."
  And I follow "MY BOOKS"
  Then I should see "The Odyssey"

Scenario: Add an existent Exemplar to My Books
  Given there is a book with title "The Odyssey" and author "Homer"
  and publisher "Sun Publications"
  Given I am signed in as user "juanlop" with password "1234"
  Given I am on the BooksOnRails home page
  And I should see a link titled "CATALOG" that goes to "/books"
  And I follow "CATALOG"
  And I press "Add to My Books"
  Then I should see "Book "The Odyssey" was already in My Books."
  And I follow "MY BOOKS"
  Then I should see "The Odyssey"
```

Con el test 8 se valida el comportamiento de la aplicación al eliminar un ejemplar de la colección personal del usuario. Se hace click en "Remove" y se comprueba que ya no se vea más el libro en la lista.

El test 8 nos hizo implementar el botón de "Remove" y el código asociado a la eliminación de un ejemplar en el controlador de ejemplares.

```
 Test 8
Feature: As a signed in user on the home page
         So that I can eliminate some exemplars from my collection
         I want to remove an exemplar

Scenario: Remove an Exemplar to My Books
  Given there is a book with title "The Odyssey" and author "Homer"
  and publisher "Sun Publications"
  Given I am signed in as user "juanlop" with password "1234"
  Given I have an exemplar of The Odyssey
  Given I am on the BooksOnRails home page
  And I should see a link titled "MY BOOKS"
  And I follow "MY BOOKS"
  And I should see "The Odyssey"
  And I press "Remove"
  And I follow "MY BOOKS"
  Then I should not see "The Odyssey"
```

Los test 9 a 13 son los que validan el comportamiento de la navegación entre las diferentes páginas mediante los enlances del menubar superior. Consisten simplemente en seguir un enlace y comprobar que se ha navegado a la página correcta.

Con los test 9 a 13, pudimos implementar una barra de menú superior funcional con posibilidad de navegación entre las distintas vistas.

```
 Test 9
Feature: As a signed in user in Home page
         So that I can see the Catalog page
         I want to navigate to the Catalog page

Scenario: Go from Home To Catalog
  Given I am signed in as user "juanlop" with password "1234"
  Given I am on the BooksOnRails home page
  And I should see a link titled "CATALOG" that goes to "/books"
  And I follow "CATALOG"
  Then I should be on the BooksOnRails catalog page
```

```
Feature: As a signed in user in Home page
         So that I can see the Account page
         I want to navigate to the Account page

Scenario: Go from Home To Account
  Given I am signed in as user "juanlop" with password "1234"
  Given I am on the BooksOnRails home page
  And I should see a link titled "ACCOUNT" that goes to "/users/1"
  And I follow "ACCOUNT"
  Then I should be on the BooksOnRails juanlop's account page
```

```
 Test 10
Feature: As a signed in user in Catalog page
         So that I can see the Home page
         I want to navigate to the Home page

Scenario: Go from Catalog To Home
  Given I am signed in as user "juanlop" with password "1234"
  Given I am on the BooksOnRails catalog page
  And I should see a link titled "HOME" that goes to "/home"
  And I follow "HOME"
  Then I should be on the BooksOnRails home page
```

```
 Test 11
Feature: As a signed in user in Catalog page
         So that I can see the Account page
         I want to navigate to the Account page

Scenario: Go from Catalog To Account
  Given I am signed in as user "juanlop" with password "1234"
  Given I am on the BooksOnRails catalog page
```

```
  And I should see a link titled "ACCOUNT" that goes to "/users/1"
  And I follow "ACCOUNT"
  Then I should be on the BooksOnRails juanlop's account page
```

```
 Test 12
Feature: As a signed in user in Account page
         So that I can see the Catalog page
         I want to navigate to the Catalog page

Scenario: Go from Account To Catalog
  Given I am signed in as user "juanlop" with password "1234"
  Given I am on the BooksOnRails home page
  And I follow "ACCOUNT"
  And I should see a link titled "CATALOG" that goes to "/books"
  And I follow "CATALOG"
  Then I should be on the BooksOnRails catalog page
```

```
 Test 13
Feature: As a signed in user in Account page
         So that I can see the Home page
         I want to navigate to the Home page

Scenario: Go from Account To Home
  Given I am signed in as user "juanlop" with password "1234"
  Given I am on the BooksOnRails juanlop's account page
  And I should see a link titled "HOME" that goes to "/home"
  And I follow "HOME"
  Then I should be on the BooksOnRails home page
```

Los test 14, 15 y 16 prueban el comportamiento de la historia de usuario de salir del sistema (logout) desde tres vistas distinas. Son simplemente un click en "LOGOUT" y comprobar que se está en la página de inicio de sesión.

Mediante los test 14, 15 y 16, implementamos la lógica del cierre de sesión en el controlador de sesiones.

```
 Test 14
Feature: As a signed in user in Home page
         So that I end the use of the site with my account
         I want to logout

Scenario: Logout from Home
  Given I am signed in as user "juanlop" with password "1234"
  Given I am on the BooksOnRails home page
  And I should see a link titled "LOGOUT" that goes to "/sessions/new"
```

```
And I follow "LOGOUT"
Then I should be on the BooksOnRails signin page
```

```
 Test 15
Feature: As a signed in user in Account page
         So that I end the use of the site with my account
         I want to logout

Scenario: Logout from Account
  Given I am signed in as user "juanlop" with password "1234"
  Given I am on the BooksOnRails juanlop's account page
  And I should see a link titled "LOGOUT" that goes to "/sessions/new"
  And I follow "LOGOUT"
  Then I should be on the BooksOnRails signin page
```

```
 Test 16
Feature: As a signed in user in Catalog page
         So that I end the use of the site with my account
         I want to logout

Scenario: Logout from Catalog
  Given I am signed in as user "juanlop" with password "1234"
  Given I am on the BooksOnRails catalog page
  And I should see a link titled "LOGOUT" that goes to "/sessions/new"
  And I follow "LOGOUT"
  Then I should be on the BooksOnRails signin page
```

4.3.2. Capturas de la aplicación

En base a los test antes presentados, se diseñaron ciertas vistas que cubrieran la validación de estos test y la lógica asociada a los controladores y modelos.

Hay que tener en cuenta que esta era una fase temprana de desarrollo por lo que simplemente interesaba la funcionalidad y no la apariencia de la aplicación. En esta iteración se desarrollaron las siguientes vistas:

La página de ingreso al sistema con un formulario simple de usuario y contraseña (ver Figura 22). Podemos ya observar el menú superior implementado con las opciones inicio de sesión y contacto.

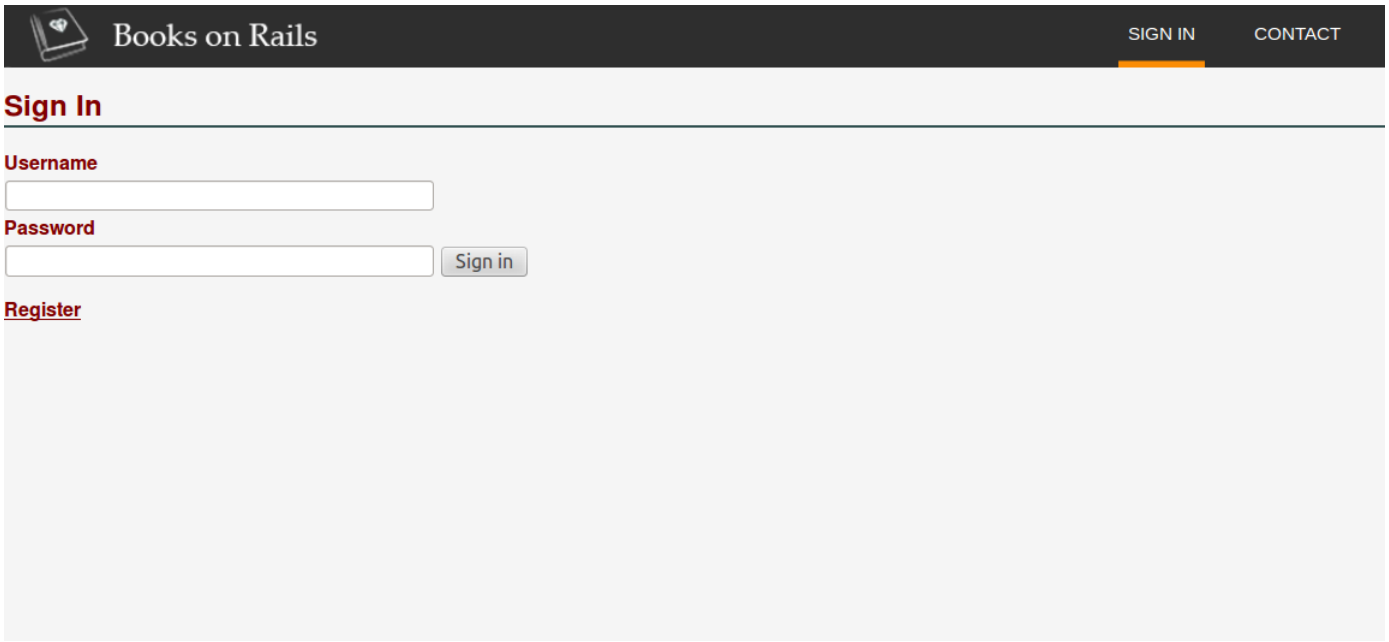

Figura 22: Página de ingreso

La página de registro de nuevos usuarios (ver Figura 23).

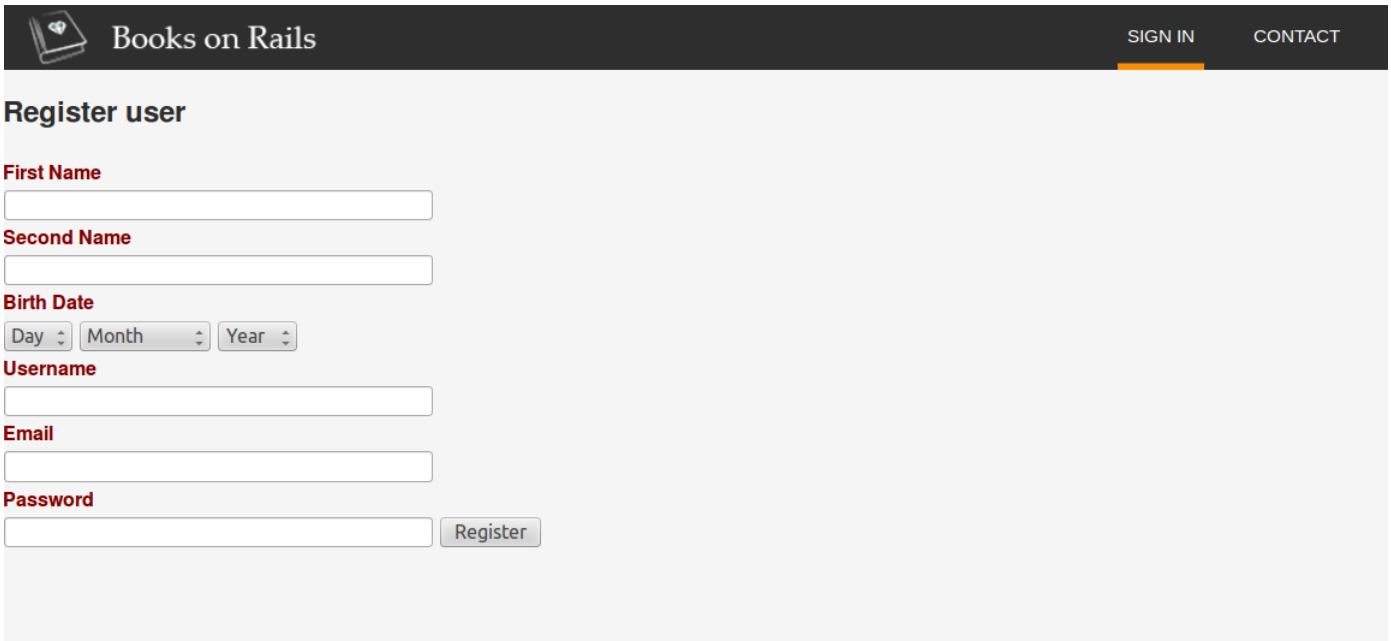

Figura 23: Registro de usuarios

La página con la información de contacto, actualmente vacía pues como hemos dicho en esta etapa del desarrollo sólo nos interesa la implementación de la funcionalidad acotada por los test de comportamiento de las historias de usuario. Será en la última iteración donde se añada dicha información de contacto. (ver Figura 24).

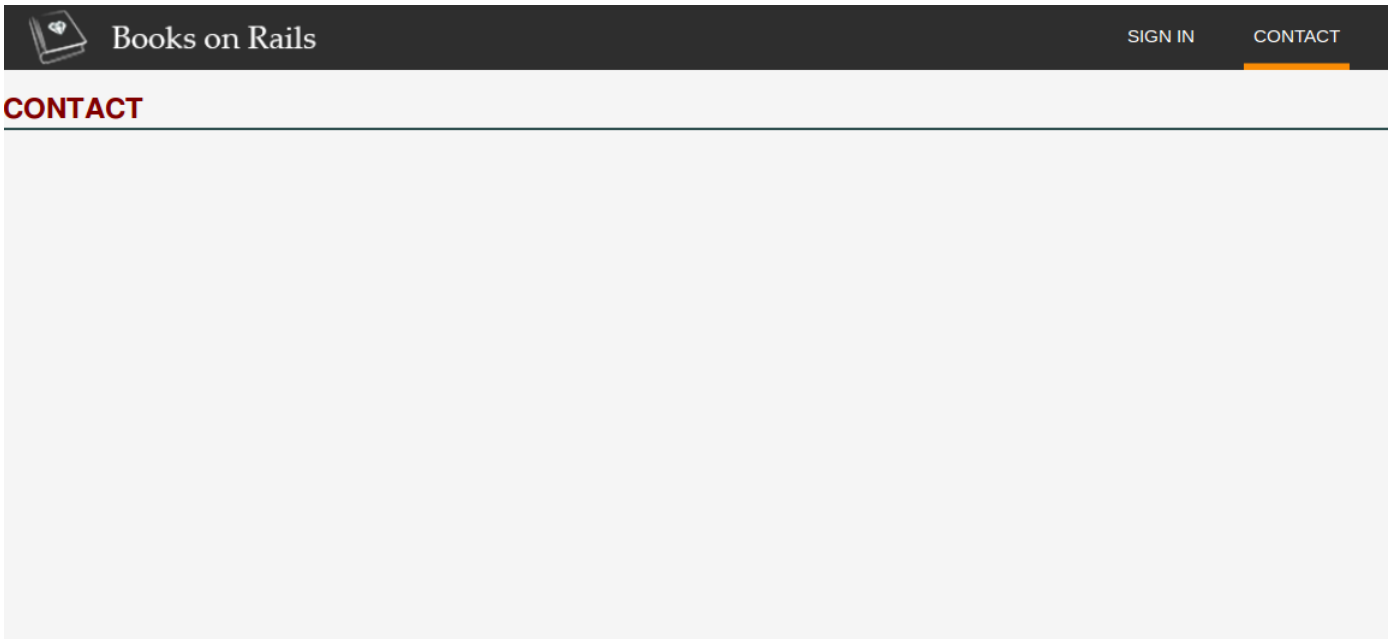

Figura 24: Página de información de contacto

La página de bienvenida, accesible nada más acceder a la aplicación con nuestra cuenta. La barra superior nos muestra ahora más opciones, visibles una vez estamos dentro de la aplicación. Con una marca naranja podemos saber en qué apartado de la aplicación nos encontramos en este mismo momento. Podemos navegar entre los distintos apartados haciendo click en su nombre del menubar (ver Figura 25).

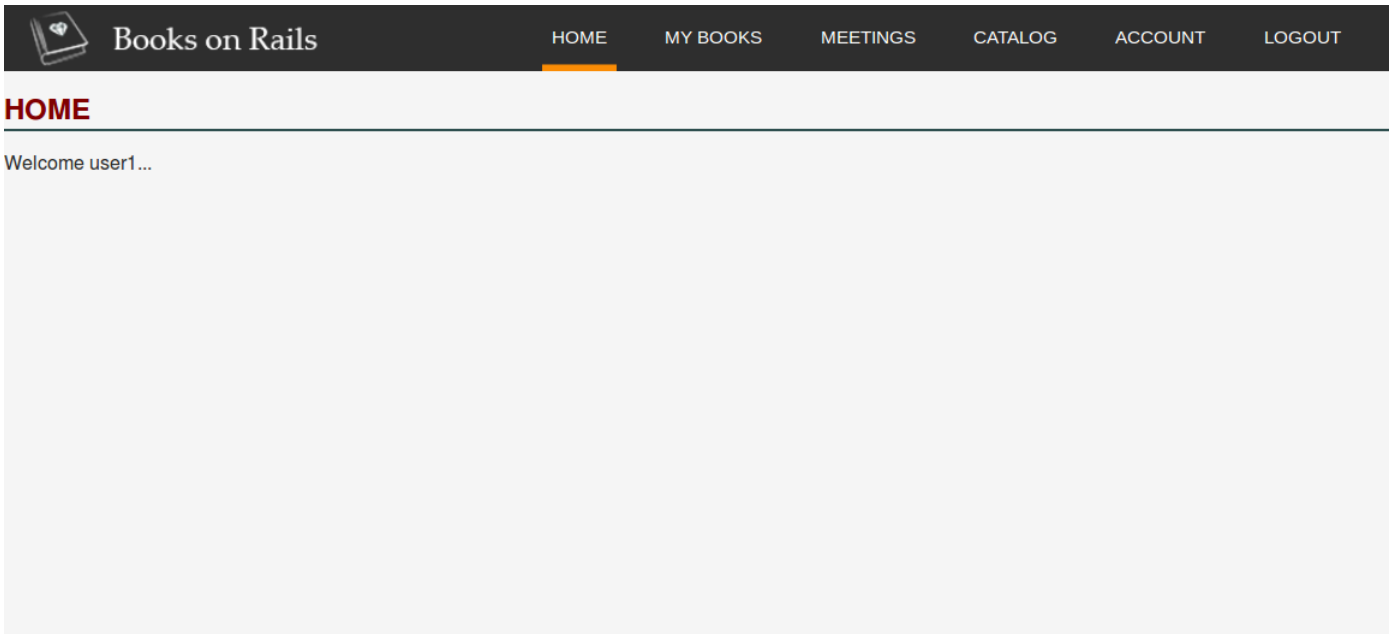

Figura 25: Página de bienvenida

La lista de libros del catálogo (en el apartado catálogo como se observa en el menubar). Podemos ver cada uno de los libros que la aplicación tiene almacenado en el catálogo general. Podemos tanto hacer click en cada uno para ver la información, añadirlos a nuestra colección o añadir un nuevo libro al catálogo (ver Figura 26).

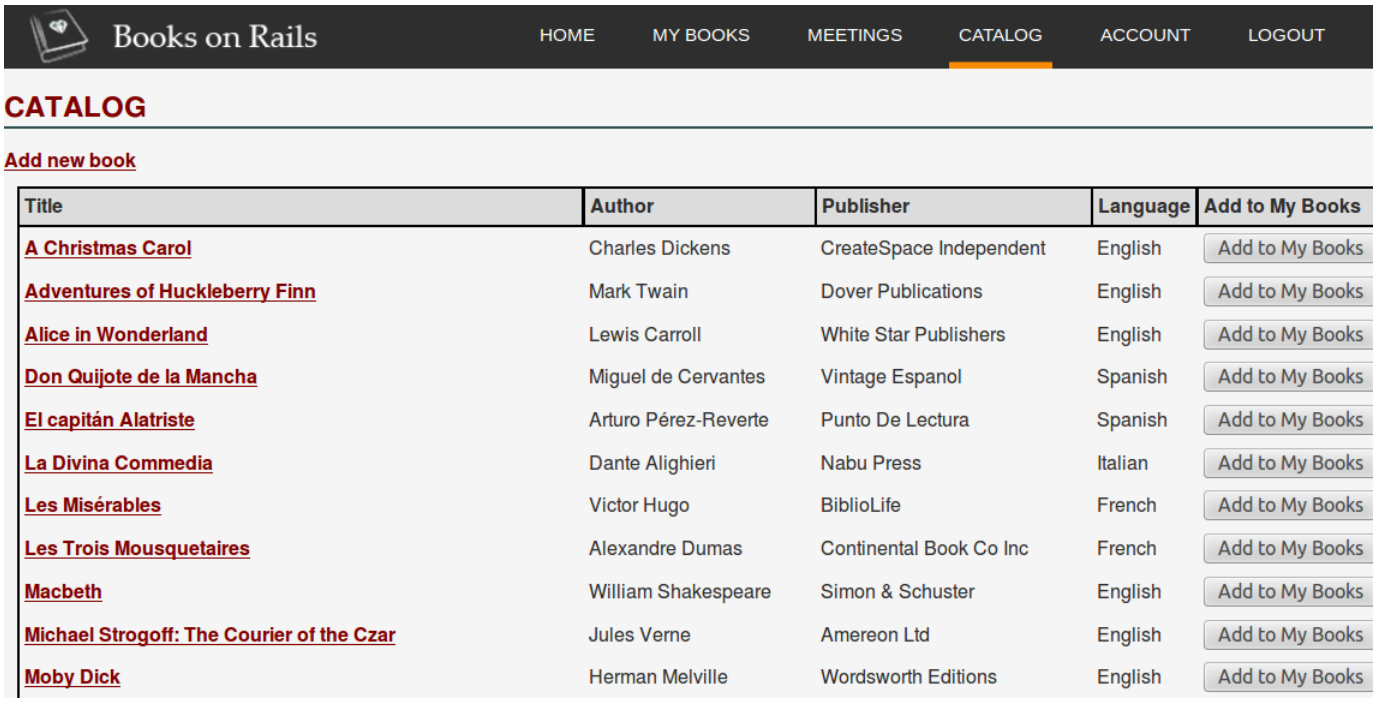

Figura 26: Catálogo de libros

Esta página muestra la vista que alberga el formulario para añadir nuevos libros al sistema (ver Figura 27).

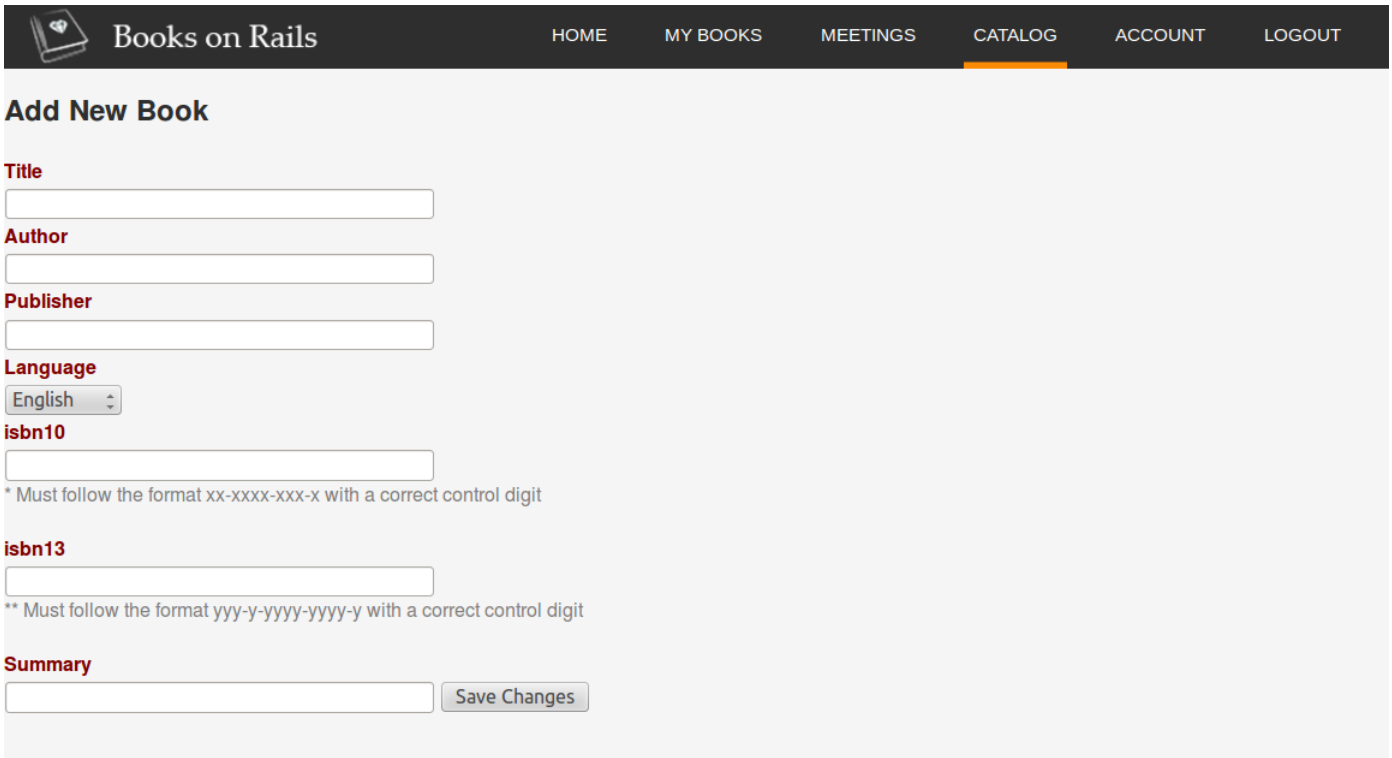

Figura 27: Añadir un nuevo libro al catálogo

Si hemos hecho click en el nombre de uno de los libros, accedemos a la página de la información en detalle de ese libro. Podemos editarla o borrar el libro del catálogo (ver Figura 28).

Figura 28: Visualizar la información de un libro

Si hemos añadido una serie de libros del catálogo general a nuestra colección, aparecerán en la lista de nuestra colección (My Books). Si no tenemos ningún ejemplar, podemos retirarlos de nuestra lista, con "Remove" (ver Figura 29).

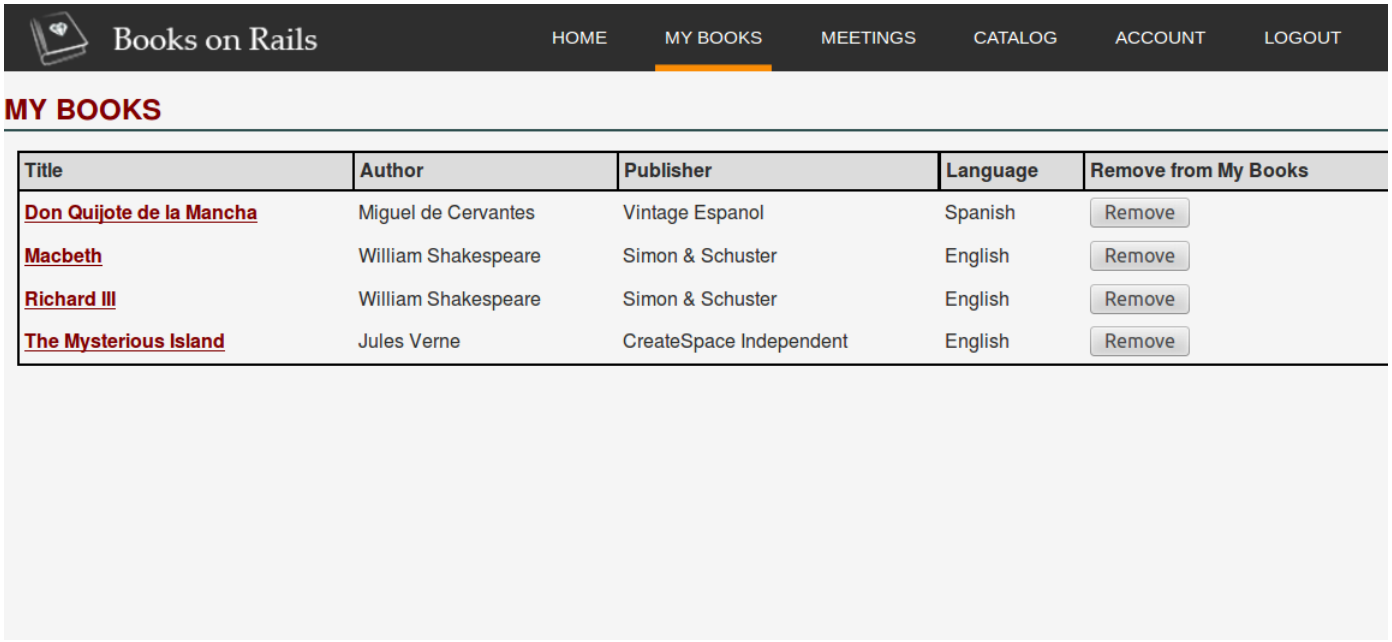

Figura 29: Colección particular de ejemplares

4.3.3. Funcionalidades implementadas

En esta iteración se ha llevado a cabo la implementación de una serie de funcionalidades, siempre precedida por un test asociado a una historia de usuario que obligue a escribir un código o a diseñar una vista que valide ese test. Las funcionalidades implementadas han sido:

- Registro de nuevos usuarios.
- Inicio y cierre de sesión con una cuenta de usuario.
- Gestión del catálogo de libros (adición, borrado, edición y visualización).
- Gestión de la colección personal de ejemplares (añadir y borrar).
- Visualización y edición de datos personales de cuenta.
- Navegación entre vistas.
- Menubar funcional.

Además en esta etapa se hizo el primer despliegue en Heroku, de manera que la web ya era accesible desde cualquier navegador, siendo este el entorno de producción que utilizaría en última instancia el cliente.

4.3.4. Funcionalidades pendientes

Estas funcionalidades son las que se ha decidido incluir en esta iteración, siendo las siguientes iteraciones en las que desarrollaremos la implementación de las demás funcionalidades y características que, tras hablar con el cliente, sabemos que faltan en la aplicación.

- Gestión de meetings (creación, edición y borrado).
- Gestión de asistencia a los meetings existentes (asistir y dejar de asistir).
- Listado de libros que llevar al meeting al que se va a asistir.
- Posibilidad de visualizar los libros que van a llevar otros usuarios.

En la siguiente iteración se llevará a cabo la implementación de las funcionalidades asociadas a meetings (reuniones) y asistencias a esas reuniones.

4.4. Iteración 2

En esta iteración se implementaron el resto de funcionalidades principales, relacionadas con la gestión de meetings y la asistencia a estos meetings.

Al estar la aplicación ya desarrollada, con una estructura y unas vistas, y habiendo superado los primeros problemas de adaptación a este nuevo framework que es Rails, esta iteración fue algo más sencilla.

4.4.1. Test de comportamiento

Todos los test de comportamiento están asociados a historias de usuario, estas historias de usuario las hemos establecido en reuniones con el cliente de manera que vamos cubriendo de forma gradual todas las funcionalidades deseadas por nuestra aplicación. En el test 17 comprobamos que se listan todos los meetings existentes en la aplicación.

Con el test 17 se implementó la página de opciones de meetings, la lista de meetings, el modelo y controlador de meetings y la lógica asociada a mostrar todos estos meetings situada en el controlador.

```
 Test 17
Feature: As a signed in user in Meetings page
         So that I can see a list of all the meetings
         I want to see a table with all the stored meetings

Scenario: View all meetings
  Given I am signed in as user "juanlop" with password "1234"
  Given there is a meeting created
  Given I am on the Meetings options page
  And I follow "View all available meetings"
  Then I should see "Meeting 1"
```

En el test 18 probamos el comportamiento de la aplicación al crear un nuevo meeting. Se rellenan los campos del formulario y se comprueba que se muestra ese meeting en la lista general de meetings.

Para validar el test 18, se ha tenido que crear la vista de creación de meetings y la lógica asociada a la creación en el controlador de meetings.

```
 Test 18
Feature: As a signed in user in Meetings page
         So that I can create a new meeting
         I want to be able to fill the data of a new meeting and create it

Scenario: Create a new meeting
  Given I am signed in as user "juanlop" with password "1234"
  Given I am on the Meetings options page
  And I follow "Create new meeting"
  And I fill in "Title" with "Meeting 2"
```

```
  And I fill in "Country" with "Spain"
  And I fill in "City" with "Madrid"
  And I fill in "Address" with "Calle Velazquez 2"
  And I press "Save Changes"
  And I should be on the all Meetings page
  Then I should see "Meeting 2"
```

En el test 19 tratamos de crear la asistencia del usuario a un meeting concreto, además diciendo que se va a llevar un libro llamado "Book 1". Se hace primero click en "Assist to meeting" en la lista de meetings, luego se añade uno de los ejemplares que el usuario tiene en su colección particular y a continuación se comprueba que efectivamente se ha creado la asistencia en la lista de meetings a los que se va a asistir.

Con el test 19, se tuvo que implementar el modelo y controlador de asistencias, la vista de nueva asistencia, la vista de lista de meetings a los que se va a asistir.

```
 Test 19
Feature: As a signed in user in Meetings page
         So that I assist to a given meeting
         I want to be able to assist and tell which books Ill go with

Scenario: View all meetings
  Given I am signed in as user "juanlop" with password "1234"
  Given there is a meeting created
  Given I have one exemplar on My Books
  Given I am on the Meetings options page
  And I follow "View all available meetings"
  And I should see "Meeting 1"
  And I follow "Assist to meeting"
  And I should see "Title: Meeting 1"
  And I should see "Book 1"
  And I press "Take book to meeting"
  Then I should see "book taken"
  And I follow "MEETINGS"
  And I follow "View meetings I'm assisting"
  Then I should see "Meeting 1"
  Then I should see "Book 1"
```

En el test 20, se trata de cancelar una asistencia a un meeting marcado como asistir. Se lista primero los meetings a los que se va a asistir, se hace click en "Remove" y OK en la ventana emergente que aparece. Se comprueba que se haya eliminado esa asistencia.

Con el test 20, implementamos el botón de eliminar asistencia y la lógica asociada a eliminar una asistencia en el controlador de asistencias.

```
 Test 20
Feature: As a signed in user in Meetings page
         So that I can stop assisting to a meeting
         I want cancel a meeting I am marked as assisting

Scenario: Cancel assistance to meetinbg
  Given I am signed in as user "juanlop" with password "1234"
  Given there is a meeting created
  Given I am assisting a meeting
  Given I am on the Meetings options page
  And I follow "View meetings I'm assisting"
  And I should see "Meeting 1"
  And I should press "Remove"
  And I click OK on the popup
  Then I should see "Assistance to 'Meeting 1' removed"
```

4.4.2. Capturas de la aplicación

De nuevo, a partir de los test anteriores se implementó código y se diseñaron vistas. Se han realizado capturas de esas vistas en esa etapa del desarrollo del proyecto.

La página que muestra la lista de meetings disponibles en toda la aplicación. Se indica si ha sido el propio usuario el que lo ha creado. Si se va a asistir se muestra en color azul, si no en amarillo (ver Figura 30).

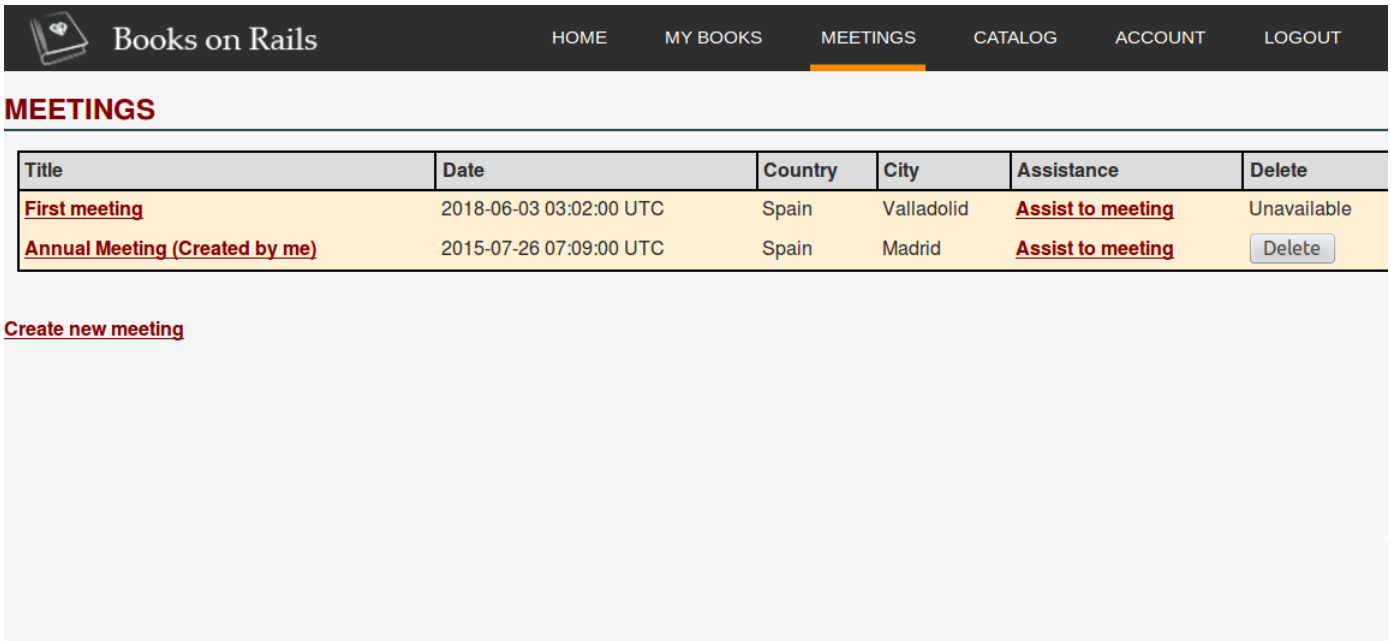

Figura 30: Lista de meetings disponibles

La creación de un nuevo meeting mediante un formulario (ver Figura 31).

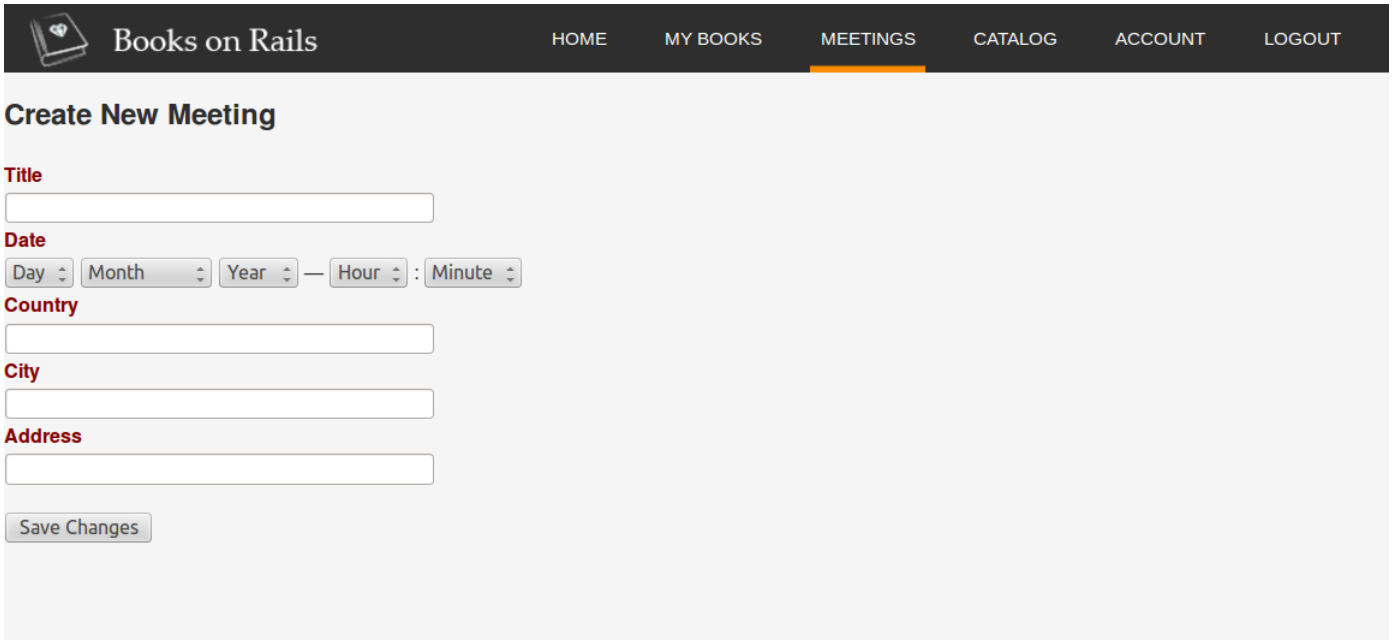

Figura 31: Creación de un nuevo meeting

Al hacer click en asistir a un meeting (en la lista de meetings) se accede a esta vista, en la que

se muestra información del meeting y los libros que se desean llevar al meeting. Si el libro se va a llevar se muestra en azul, si no en amarillo (ver Figura 32).

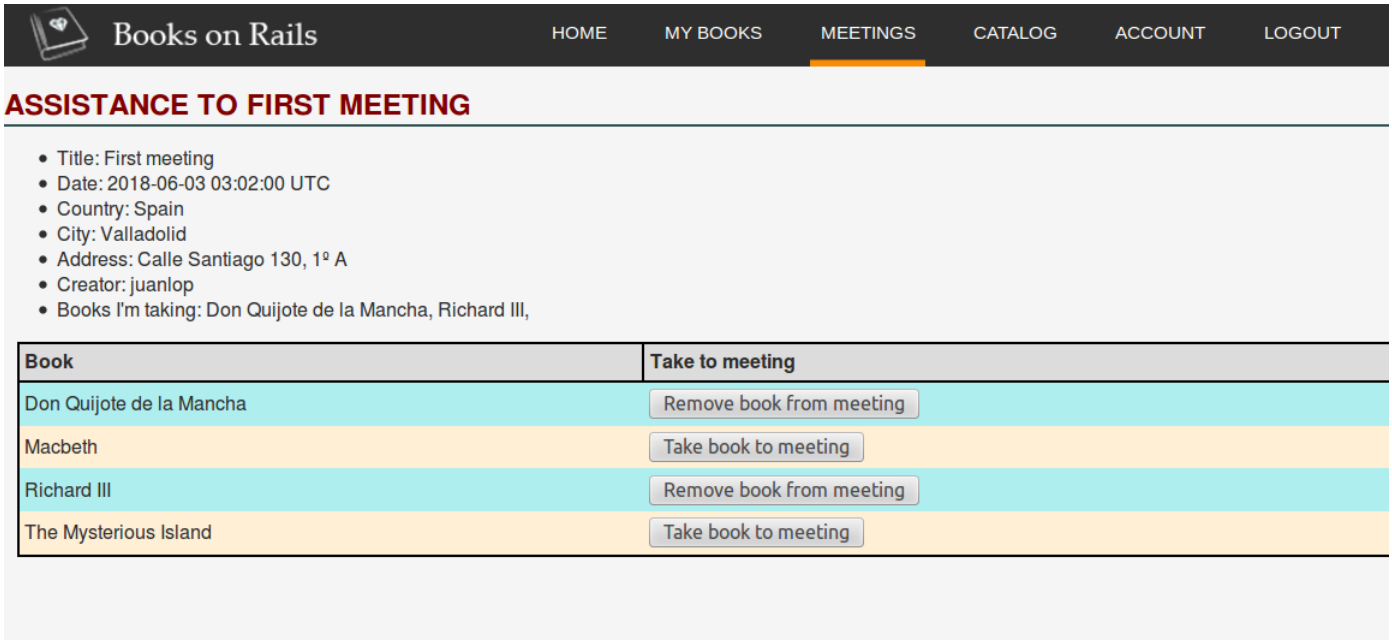

Figura 32: Assistiendo a un meeting

Si en la lista de meetings se hace click sobre el nombre de uno de los listados, se accede a la información del meeting de manera que el usuario puede ver qué gente va a asistir o si le interesa ir (ver Figura 33).

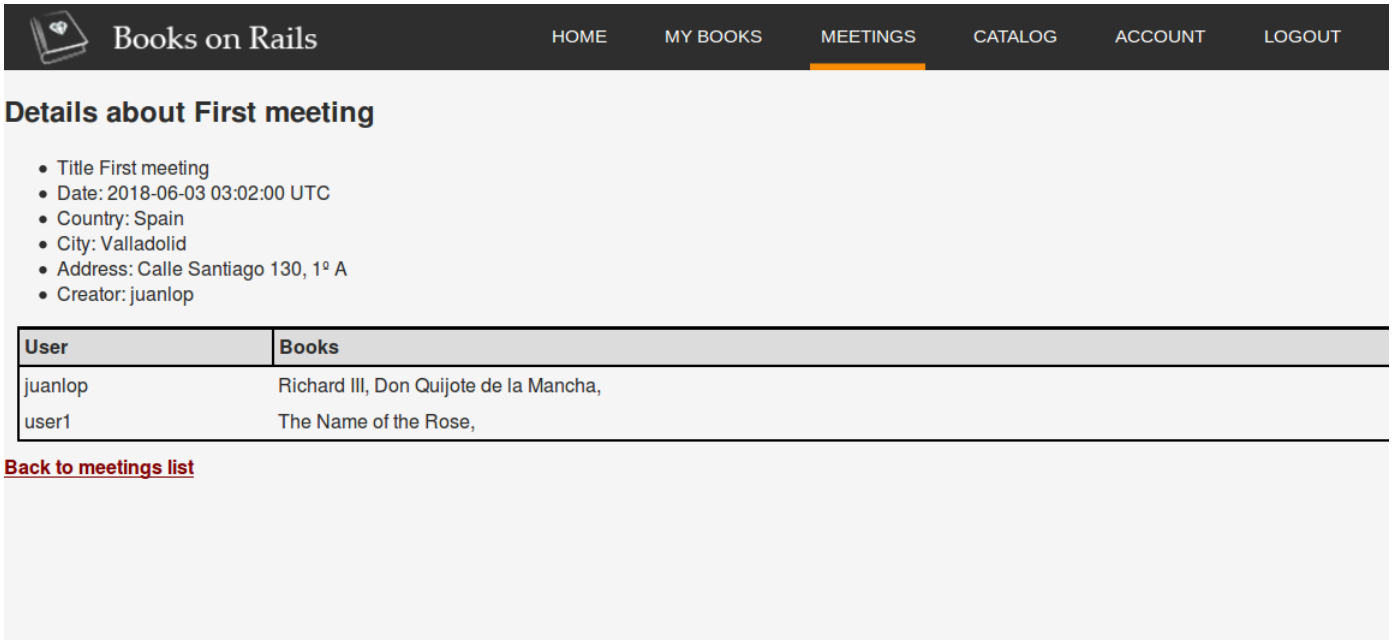

Figura 33: Información de un meeting

Aquí se listan los meetings a los que el usuario ha marcado como asistir (ver Figura 34).

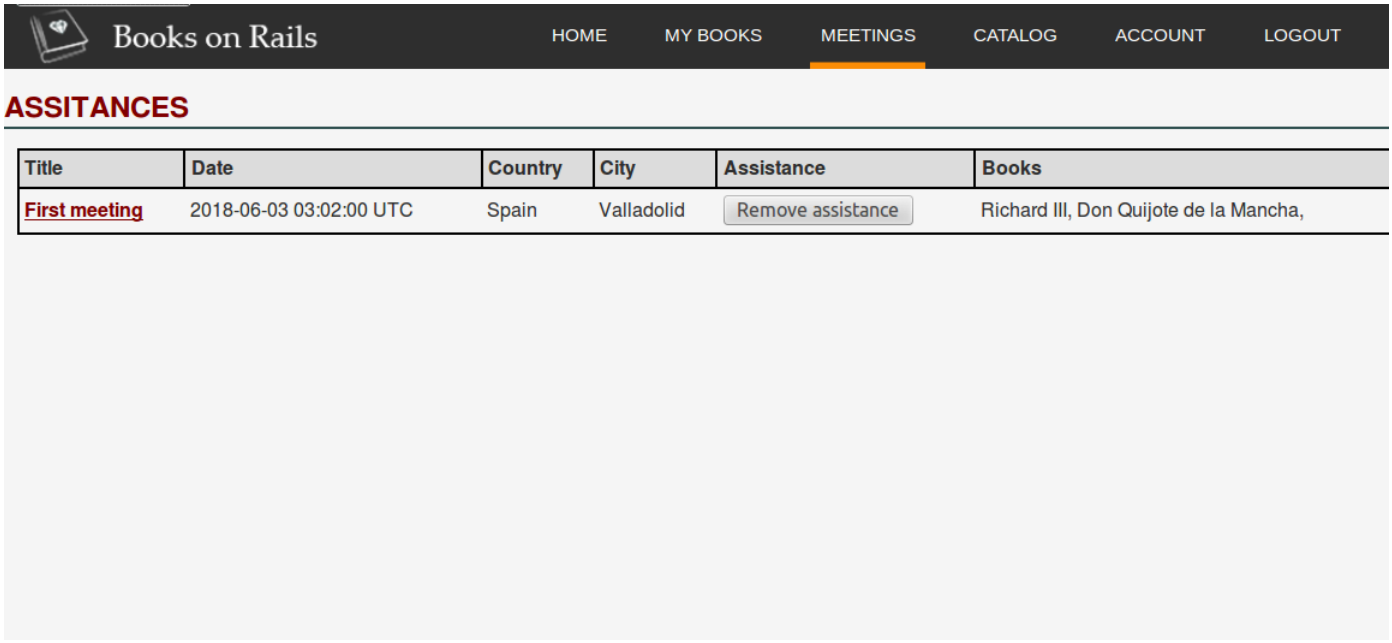

Figura 34: Lista de meetings a los que va a asistir

4.4.3. Funcionalidades implementadas

En esta iteración se cubren las funcionalidades principales de la aplicación de manera general.

- Gestión de meetings (creación, borrado, edición y visualización).
- Gestión de asistencia a los meetings existentes (asistir y dejar de asistir).
- Listado de libros que llevar al meeting al que se va a asistir.
- Posibilidad de visualizar los libros que van a llevar otros usuarios.

Además se vuelve a desplegar la aplicación en Heroku para comprobar que el entorno de producción sigue perfectamente estable y funcional.

4.4.4. Funcionalidades y características pendientes

Aunque ya hemos cubierto todas las funcionalidades planteadas por la aplicación, no está preparada para salir a producción en forma de release. Faltan algunas features menores y un lavado de cara en forma de formato y estilos de diseño web.

- Búsqueda de libros y adición de ejemplares desde My Books
- Control de códigos Isbn10 y Isbn13
- Configurar ciertos aspectos de la fecha de nacimiento del usuario
- Apariencia óptima y final de la web.

4.5. Iteración 3

Esta es la última iteración, es el momento en que capturamos el estado final de la aplicación, con todas las funcionalidades implementadas y la apariencia definitiva. Esta versión producida es la que se considera como primera release, disponible en entorno de producción.

4.5.1. Test de comportamiento

En este caso los test de comportamiento han sido mínimos pues en realidad nuevas funcionalidades no hemos añadido más que una. Este test corresponde a la historia de usuario de buscar un libro desde "My Books" en función del título y que se puedan añadir libros desde aquí. Se rellena el cuadro de texto con la cadena que queramos que contenga el título. Se mostrará una lista de libros que concuerden con esos términos de búsqueda. Se pulsa "Add to My Books" para añadirlo a la colección propia.

Este test hizo que se tuvieran que implementar elementos extra en la vista de "My Books" para mostrar la parte correspondiente a la búsqueda. Además toda la lógica de búsqueda en el controlador de ejemplares.

```
Feature: As a signed in user in My Books page
         So that I can quickly add an exemplar to my own collection
         I want to be able to search and add books directly from My Books page


Scenario: Search and add from My Books
  Given there is a book with title "The Odyssey" and author "Homer"
  and publisher "Sun Publications"
  Given I am signed in as user "juanlop" with password "1234"
  Given I am on the BooksOnRails home page
  And I follow "MY BOOKS"
  And I fill in Searchform with "The Odyssey"
  And I press "Search"
  The I should see "The Odyssey"
  And I press "Add to My Books"
  Then I should see "Book "The Odyssey" has been added to My Books."
```

4.5.2. Capturas de la aplicación

En este caso se muestran las capturas con el estado final de apariencia de la aplicación. No se muestra más que las nuevas funcionalidades añadidas, esto es la búsqueda de libros desde la colección personal de ejemplares.

En la página de la colección de libros tenemos un cuadro de texto para buscar libros por título. En este caso de ejemplo se ha buscado una palabra y se muestran los títulos que coinciden con ella. Desde aquí se puede además añadir un ejemplar de ese libro a la colección (ver Figura 35).

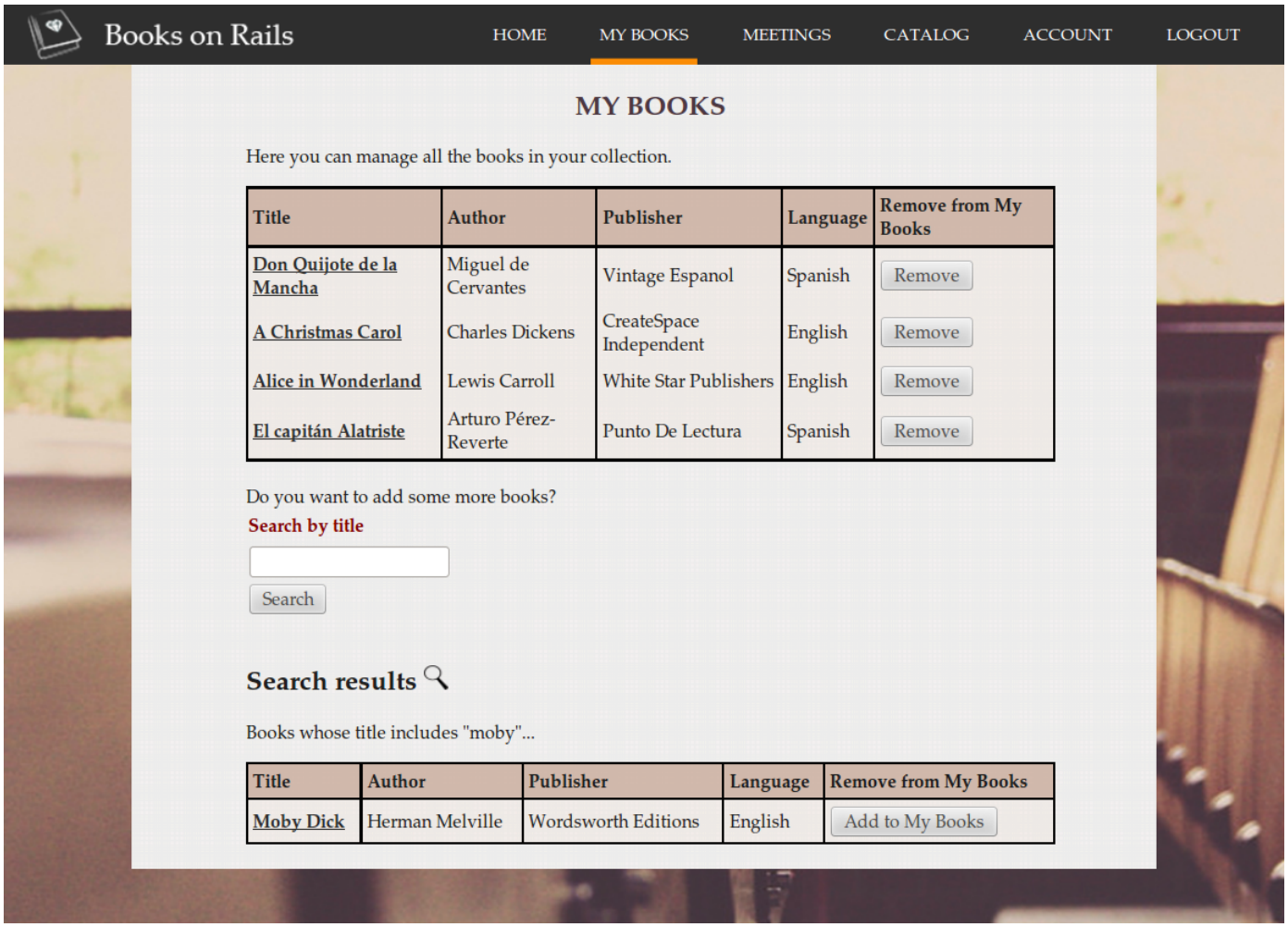

Figura 35: Búsqueda de libros desde la colección personal

Si se ha optado por añadir ese libro a la colección personal, se muestra un mensaje en verde alertando con la acción ocurrida. (ver Figura 36).

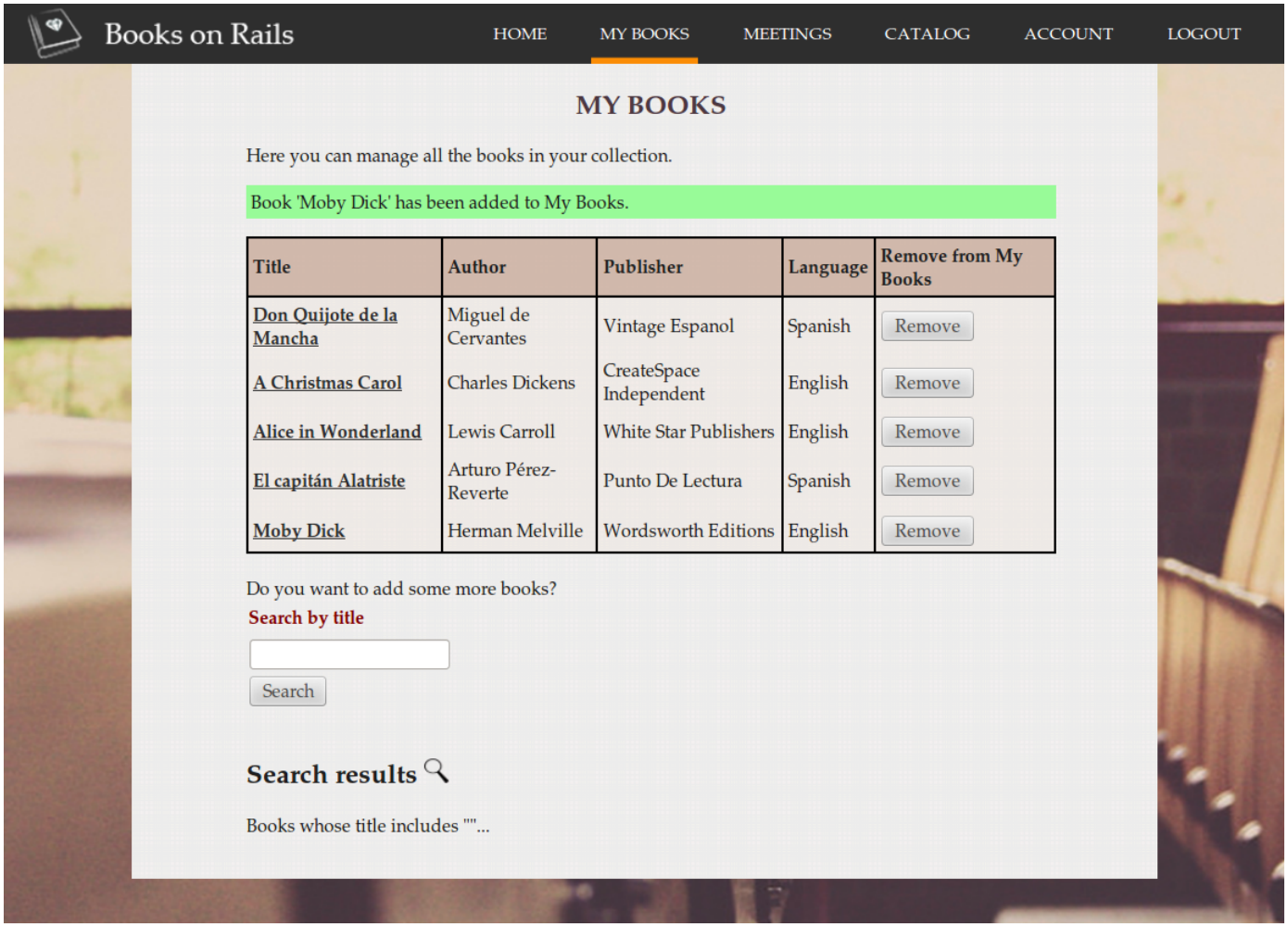

Figura 36: Añadir un ejemplar desde la colección personal

4.5.3. Control de códigos ISBN10 y ISBN13

Otra funcionalidad no tan obvia que se ha añadido en esta iteración es el control de códigos ISBN. Los códigos ISBN son utilizados por las librerías y editoriales como identificador único para los libros. Antiguamente existía únicamente el ISBN de diez dígitos (ISBN10) y más adelante surgió el de trece (ISBN13) para tener la posibilidad de identificar mayor número de libros.

Estos códigos suelen estar separados por guiones. En el caso del ISBN13 no existe un formato de guiones estándar, por lo que hemos tomado el más común como el correcto. De esta manera la distribución debería ser así:

- **ISBN10**: 2 cifras - 4 cifras - 3 cifras - 1 cifra (ej: 84-6632-053-9)
- **ISBN13**: 3 cifras - 1 cifra - 4 cifras - 4 cifras - 1 cifra (ej: 978-8-4663-2053-5)

  El último dígito de ambos códigos es un dígito de control, calculado en función del resto. En el caso del ISBN10 basado en el módulo 11 y en el del ISBN13 en el módulo 10.

4.5.4. Apariencia de la aplicación

Al ser la última iteración, se ha aplicado la apariencia definitiva de la aplicación. Por eliminar redundancia entre las capturas de la apariencia y el manual de la aplicación, utilizaremos este último para mostrar el estado definitivo de la aplicación.

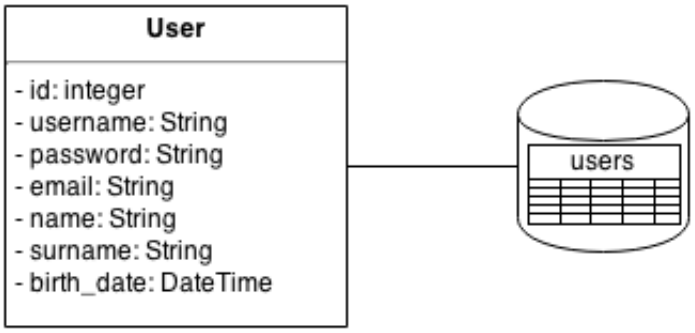

Figura 37: Tabla y modelo de usuarios

4.6. Estado final de la implementación

4.6.1. Tablas y restricciones de los modelos

Como ya hemos explicado, en Ruby on Rails, cada tabla tiene asociado una clase de la capa del Modelo (recordamos que es MVC) que posee tantos atributos como columnas tenga la tabla. Estos modelos tienen una serie de restricciones como el que un valor de un atributo sea único en el sistema. Todos ellos tienen como restricción la exclusividad del valor del id.

En este proyecto se han implementado las siguientes tablas y modelos con unas restricciones asociadas a cada uno.

4.6.1.1. Users

La Figura 37 representa al modelo y la tabla correspondientes a los usuarios.

Las restricciones que posee este modelo son (aparte de la exclusividad de id):

- Obligatoriedad de username, email y password.
- Exclusividad de username y email.

4.6.1.2. Sessions

La Figura 38 representa al modelo y la tabla correspondientes a las sesiones de usuario asociadas a cada acceso a la aplicación con una cuenta de usuario.

Este modelo no posee restricciones aparte de la exclusividad de id.

4.6.1.3. Books

La Figura 39 representa al modelo y la tabla correspondientes a los libros almacenados en la aplicación.

Las restricciones que posee este modelo son (aparte de la exclusividad de id):

- Obligatoriedad de title, author, isbn10 e isbn13.
- Exclusividad de isbn10 e isbn13 (ambos de manera simultánea).
- Longitud de isbn10 (13 caracteres, con guiones) e isbn13(17 caracteres, con guiones).

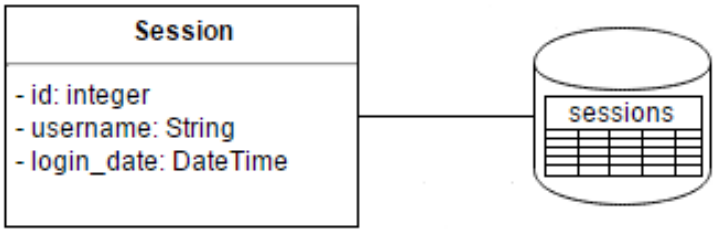

Figura 38: Tabla y modelo de sesiones

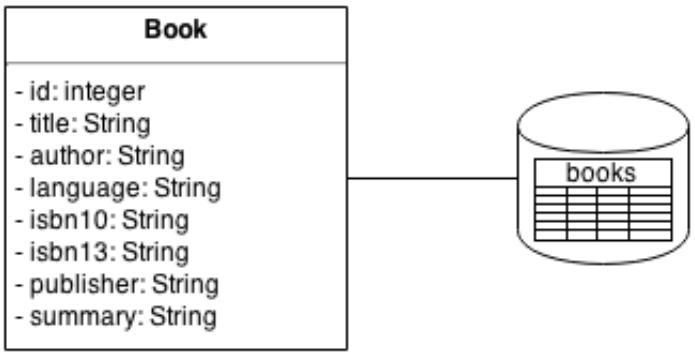

Figura 39: Tabla y modelo de libros

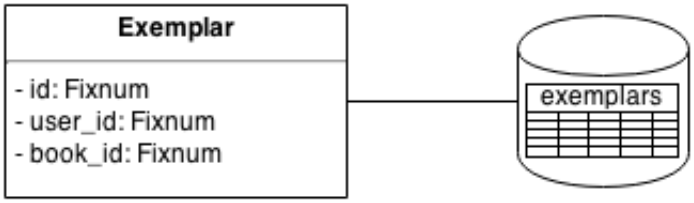

Figura 40: Tabla y modelo de ejemplares

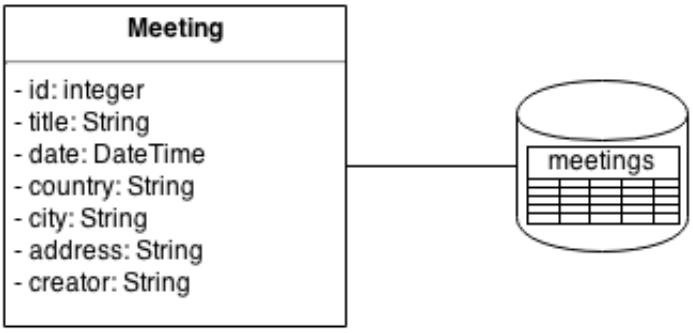

Figura 41: Tabla y modelo de meetings

- Tanto isbn10 como isbn13 deben adecuarse a un formato determinado(ver 4.5.3, Control de códigos ISBN10 y ISBN13).

4.6.1.4. Exemplars

La Figura 40 representa al modelo y la tabla correspondientes a los ejemplares particulares de los libros ya existentes.

Las restricciones que posee este modelo son (aparte de la exclusividad de id):

- Obligatoriedad de user_id y book_id.
- Exclusividad de la combinación user_id-book_id.
- Ambos id deben apuntar a entradas válidas de la base de datos.

4.6.1.5. Meetings

La Figura 41 representa al modelo y la tabla correspondientes a los meetings almacenados en la aplicación.

Las restricciones que posee este modelo son (aparte de la exclusividad de id):

- Obligatoriedad de title.
- Exclusividad de title.

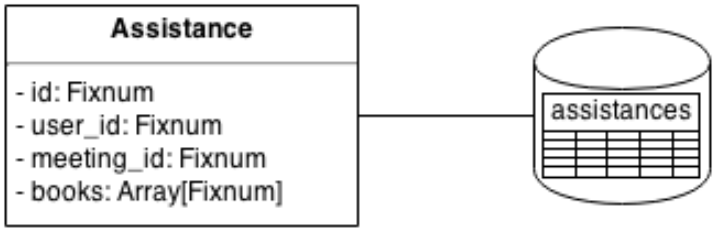

Figura 42: Tabla y modelo de asistencias

4.6.1.6. Assistances

La Figura 42 representa al modelo y la tabla correspondientes a las asistencias de usuarios a meetings existentes en el sistema.

Las restricciones que posee este modelo son (aparte de la exclusividad de id):

- Obligatoriedad de user_id y meeting_id.
- Exclusividad de la combinación user_id-meeting_id.
- Ambos id deben apuntar a entradas válidas de la base de datos.

4.6.2. Controladores

En el apartado dedicado a MVC (ver 3.2.2) se explicaba que el controlador tomaba el papel de intermediario entre vistas y modelos. En el caso de Ruby on Rails, se necesita un controlador por modelo y dentro de cada uno, un método por cada una de las funciones que se realicen sobre el modelo asociado.

En la Figura 43 podemos observar todos los controladores implementados, uno por cada clase de la capa del modelo y uno extra (MainPagesController) que gestiona las páginas no pertenecientes a un modelo, como es el caso de la página de información de contacto.

Cada clase implementa unos métodos (new(), create(), index(), etc) relacionados directamente con ciertas vistas y otros métodos específicos para cierta lógica añadida (add_to_my_books(), sign_to_meeting()).

4.6.3. Vistas

Los elementos de la capa de la Vista se corresponden con las capturas que hemos ido mostrando con anterioridad por lo que no tiene sentido volverlos a mostrar.

En el diagrama de relaciones (ver 4.6.4) y en la estructura de carpetas del proyecto (ver 4.6.5) se muestra una relación de todas las vistas diseñadas en la aplicación.

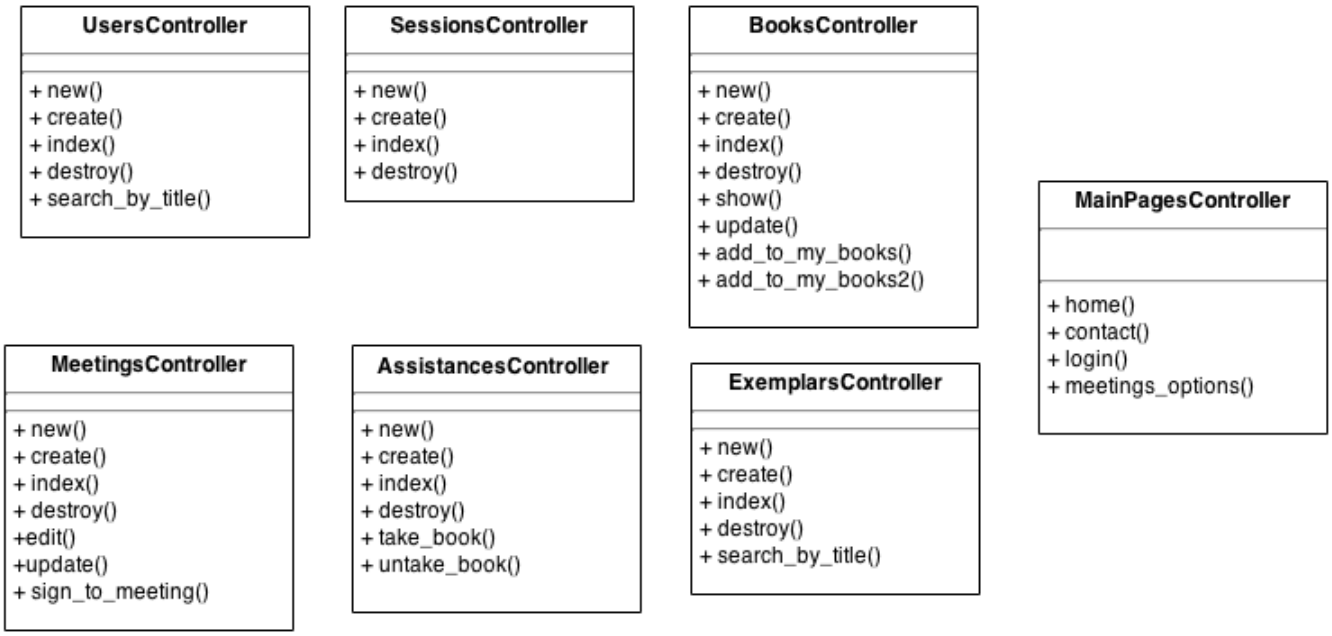

Figura 43: Clases de la capa Controlador de la aplicación

4.6.4. Diagrama final de la implentación

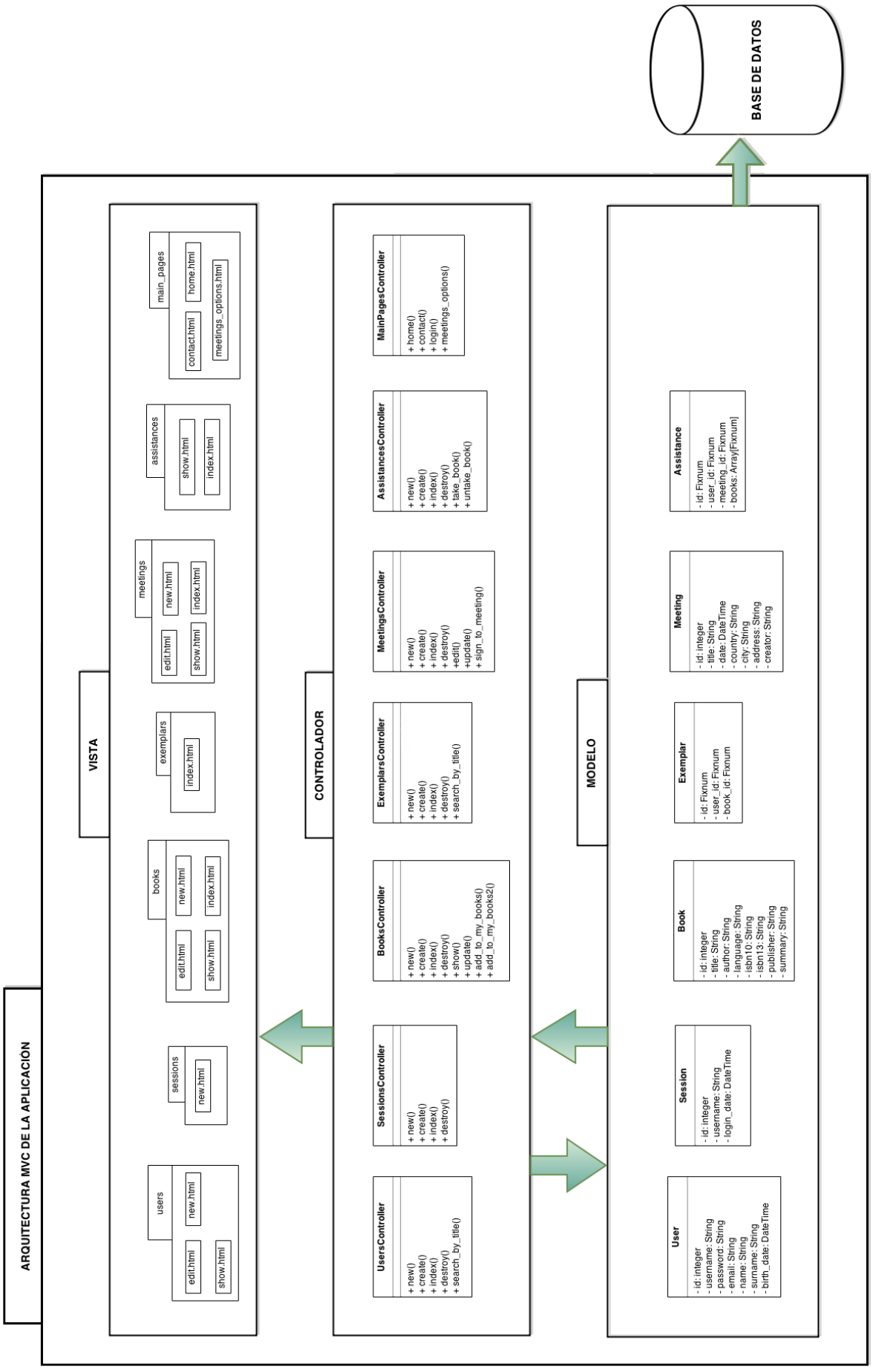

Figura 44: Diagrama de la implementación

4.6.5. Estructura de carpetas

Como hemos visto, Ruby on Rails tiene una manera muy concreta de trabajar y estructurarse, basándose en convenciones como nombres determinados o carpetas en las que deben ir ciertos elementos de la aplicación.

Aquí presentamos la estructura de carpetas que se ha generado en este proyecto al utilizar Ruby on Rails.

```
app/
      assets/
        images/
            backgroundLogo.png
            bg.jpg
            pat3.png
            search.png
            topMenuImages.png
        javascripts/
        stylesheets/
            application.css
    controlers/
        application_controller.rb
        assistances_ controller.rb
        books_controller.rb
        exemplars_controller.rb
        main_pages_controller.rb
        meetings_controller
        sessions_controller.rb
        users_controller.rb
    helpers/
        application_helper.rb
    models/
        assistance.rb
        book.rb
        exemplar.rb
        meeting.rb
        session.rb
        user.rb
    views/
        assistances/
            index.html.haml
            show.html.haml
        books/
            _book_form.html.haml
            edit.html.haml
            index.html.haml
```

```
            new.html.haml
            show.html.haml
        exemplars/
            index.html.haml
            search_by_title.html.haml
        layouts/
            application.html.haml
        main_pages/
            contact.html.haml
            home.html.haml
            meetings_options.html.haml
        meetings/
            _meeting_form.html.haml
            edit.html.haml
            index.html.haml
            new.html.haml
            show.html.haml
        menubars/
            _menubar_account.html.haml
            _menubar_catalog.html.haml
            _menubar_contact.html.haml
            _menubar_home.html.haml
            _menubar_meetings.html.haml
            _menubar_mybooks.html.haml
            _menubar_signgin.html.haml
        sessions/
            new.html.haml
        users/
            _user_form.html.haml
            edit.html.haml
            new.html.haml
            show.html.haml
config/
    enviroments/
        development.rb
        production.rb
        test.rb
    application.rb
    boot.rb
    cucumber.rb
    database.yml
    enviroment.rb
    routes.rb
```

```
db/
    migrate/
        20141114120246_create_books.rb
        20141117123909_add_language_to_books.rb
        20141118183945_create_users.rb
        20141119111356_create_sessions.rb
        20141127105539_create_exemplars.rb
        20141203173637_create_meetings.rb
        20141205162008_change_data_type_for_meeting_date.rb
        20141206193542_create_assistances.rb
    schema.rb
    seeds.rb
features/
    step_definitions/
        web_steps.rb
    support/
        paths.rb
    *{Cada una de la features}*.feature
log/
    development
Gemfile
```

4.7. Cálculo final del presupuesto

En el apartado 4.7: Cálculo inicial del presupuesto, calculábamos el coste a priori de este proyecto software. Ahora vamos a analizar el coste real, compararlo con el anterior y sacar conclusiones del resultado.

El coste de la hora de trabajo se mantiene en los 5 euros que habíamos supuesto.

El número de horas empleadas en el desarrollo del proyecto se ha calculado del siguiente modo. El número de días trabajados ha sido del 1 de Octubre al 10 de Enero (100 días) a una media de 4.5 horas diarias, teniendo en cuenta días en los que no se ha trabajado y días en los que se ha superado el número medio de horas.

Por tanto el coste final del desarollo del proyecto software sería:

```
Número de horas: 450 horas
Remuneración del desarrollador: 5 euros/hora

300 horas * 5 euros/hora = 2250 euros
```

Esto supone una desviación del 50 % sobre el presupuesto inicial (1500 euros). Es decir, se han gastado 750 euros de más, empleados en 150 horas de trabajo extra.

Si valoramos este resultado sacamos la conclusión de que el coste añadido sobre el presupuesto incial es debido a:

- Desconocimiento del lenguaje y framework a implementar.
- Inexperiencia en desarrollo con metodología ágil.
- Inexperiencia en BDD (Behaviour Driven Design).
- Documento de seguimiento del proyecto con una estructura no tradicional.

El exceso de horas está acumulado al principio del desarrollo del proyecto, cuando no se conocía el lenguaje o la metodología y cada pequeño paso implicaba el aprendizaje de nuevos conceptos o formas de trabajar. A medida que se fue avanzando con el desarrollo de la aplicación se fue ganando velocidad y seguridad. Al final se alcanzó una velocidad de producción superior a la de un desarrollo convencional, sin llegar a amortiguar el coste extra necesario al principio del desarrollo.

Cabe por tanto suponer que si se desarrollasen más proyectos utilizando mismo procedimiento y tecnologías, su coste sería reducido respecto a un desarrollo tradicional.

5 Conclusiones

5.1. Objetivos cumplidos

Al principio de esta memoria, se explicaban los objetivos iniciales que se pretendían alcanzar con el desarrollo de este proyecto. Al final del desarrollo hacemos un pequeño análisis y listamos los objetivos cumplidos y otros objetivos conseguidos paralelos conseguidos durante el desarrollo.

- Aprendizaje de un nuevo lenguaje (Ruby).
- Aprendizaje e implementación de un framework avanzado (Ruby on Rails).
- Desarrollo de un proyecto con una metodología ágil.
- Avanzar en ese desarrollo mediante BDD (Behaviour-Driven-Design).
- Despliegue de una aplicación en la nube, en un entorno de producción.
- Aprendizaje y utilización de una serie de herramientas de ayuda tanto a la planificación como al desarrollo.
- Diseño completo de una página web.
- Tomar el papel de cliente en el desarrollo de un proyecto software.
- Desarrollo de un documento de especificación y seguimiento del proyecto.

5.2. Propuestas de mejora

El estado final del proyecto es el que nos encontramos en la iteración 3, pero no quiere decir que no pueda ser mejorado o ampliado con nuevas funcionalidades o características.

Por diversos motivos, ya sea intentar controlar el alcance del proyecto o no incrementar su complejidad, ciertas funcionalidades no se han implementado pero si se han dejado planteadas como mejoras en potencia.

5.2.1. Autenticación Single Sign-On (SSO)

Este tipo de autenticación (acceso a la aplicación con una cuenta) es un sistema centralizado de autorización ofrecido por third-parties (empresas particulares externas) como es el caso de Facebook o Twitter.

Con una cuenta en una de estas redes sociales podemos crearnos una cuenta en nuestra aplicación con un solo click. Es un sistema muy extendido por su simplicidad, su rapidez y su probada seguridad pues no genera mayores problemas que crearse una cuenta en la propia aplicación.

En Rails este sistema está soportado por un módulo (gema) llamado OmniAuth.

Aunque en etapas tempranas se probó a realizar una SSO en una aplicación de prueba con una cuenta de Twitter, al final se desechó la idea por complejidad y la existencia de una fecha de entrega.

5.2.2. Responsive Design

Otra posible característica sería diseño adaptativo, es decir que en función de la resolución de la pantalla, los elementos en esta se recoloquen ofreciendo una interfaz accesible y cómoda por ejemplo desde dispositivos móviles.

5.2.3. Alimentación del catálogo de libros via API

A pesar de que primero fue pensado como una característica básica que ofrecería nuestra aplicación, debido a las restricciones de tiempo para el desarrollo del proyecto y la complejidad que acarrea la implementación de esta funcionalidad, se decidió no implementarla. Se explica ahora como una característica deseable para nuestra aplicación en un futuro.

El uso de una API externa conectada a una base de datos de libros con la que poder alimentar nuestro catálogo u ofrecer un sistema de búsqueda que devuelva resultados de su base de datos es una excelente característica.

Este sistema está soportado por un módulo desarrollado por un particular, disponible en github: `https://github.com/sethvargo/isbndb/`.

La API ofrecida por `www.isbndb.com` es de uso abierto, solamente hace falta una clave de acceso que te proporcionan ellos con capacidad de hasta 500 consulta s gratuitas diarias.

5.2.4. Aplicación móvil

Otra posible característica hubiera sido el desarrollo de una aplicación móvil que consumiese los servicios de Rails, recibiese información JSON o XML y la mostrase en una aplicación Android, iOS o Windows Phone, y a la vez enviase acciones, tal como hace la apliación web accesible desde navegador.

5.2.5. Geolocalización

Para situar un nuevo meeting en nuestra aplicación debemos indicar País, Ciudad y Dirección. Una opción sería utilizar una herramienta como Google Maps para situar el meeting en un mapa.

5.2.6. Conexión con social media

Algo que está muy de moda es dotar a todas las aplicaciones de un aspecto social, nuestra aplicación tiene ya un componente social implícito, lo que habríamos de hacer sería conectarla con las redes más usadas actualmente.

Podríamos añadir opciones de "Me gusta" o compartir (en el muro de Facebook o el timeline de Twitter). De esta manera probablemente crecería el uso que la gente hace de nuestra aplicación.

5.2.7. Herramienta de comunicación interna

Se podría tambien implementar una comunicación entre usuarios de manera directa en forma de chat o de mensajes. Además mantener un registro de amigos de tal manera que sea más fácil comunicarse con usuarios con los que te comunicas frecuentemente.

5.3. Valoraciones y conclusiones finales

A partir del desarrollo de un proyecto de estas características tanto a nivel de lenguaje de programación o framework como de metodología de desarrollo se han sacado una serie de conclusiones y valoraciones.

Al tratarse de un lenguaje nuevo evidentemente se requiere un esfuerzo mayor que si se hubiese decidido implementar en un lenguaje conocido. En cuanto al framework, en la carrera no se hace especial hincapié en el aprendizaje de ningún framework debido a su complejidad y al tiempo necesario para conocerlo en profundidad, por lo que el uso de Ruby on Rails fue al principio algo tedioso.

No es habitual en ninguna asignatura encontrarse con metodología ágil, siempre se suele seguir el ciclo Análisis-Diseño-Implementación-Pruebas de UP (Proceso unificado), por lo que aunque se reduce la cantidad de documentación y comunicación escrita necesaria, se necesita un esfuerzo extra para seguir un plan de desarrollo por iteraciones e historias de usuario. Además, al desarrollar con BDD, el hecho de tener que escribir un test antes de escribir nada de código, ralentiza el avance del proyecto, pues no es la manera "natural" de programar.

Las valoración que se ha sacado del proyecto es enormemente positiva. Aunque haya supuesto un mayor esfuerzo y tiempo por tratarse como hemos dicho de tecnologías y metodologías concretas no tradicionales, se ha obtenido un conocimiento y experiencia que de otra forma no hubiera sido posible.

Gracias a la supervisión de la profesora Yania Crespo González-Carvajal se han seguido una serie de pautas y recomendaciones sobre el desarrollo del proyecto software y el documento de seguimiento del mismo, que han sido de gran ayuda.

Referencias

[1] David Patterson Armando Fox. Engineering software as a service: An agile approach using cloud computing. Última visita el 10/12/2014.

[2] UC BerkeleyX. Engineering software as a service. https://www.edx.org/course/engineering-software-service-uc-berkeleyx-cs169-1x#.VIwqDyuG_pA. Última visita el 10/12/2014.

[3] UC BerkeleyX. Engineering software as a service, part 2. https://www.edx.org/course/engineering-software-service-part-2-uc-berkeleyx-cs169-2x#.VIwqECuG_pA. Última visita el 10/12/2014.

[4] Steve Mellor Jim Highsmith and 15 others. The agile manifesto. http://agilemanifesto.org/authors.html, 2001. Última visita el 02/12/2014.

[5] American Society for Quality. Continuous improvement. http://asq.org/learn-about-quality/continuous-improvement/overview/overview.html. Última visita el 9/12/2014.

[6] International Software Testing Qualifications Board. What is agile model – advantages, disadvantages and when to use it? http://istqbexamcertification.com/what-is-agile-model-advantages-disadvantages-and-when-to-use-it/. Última visita el 12/12/2014.

[7] Tdd. test driven development. http://en.wikipedia.org/wiki/File:Test-driven_development-UML.PNG. Última visita el 8/01/2015.

[8] BDDWiki. Behaviour driven development. http://behaviour-driven.org/. Última visita el 14/12/2014.

[9] BDDWiki. Behaviour driven development introduction. http://behaviour-driven.org/Introduction. Última visita el 14/12/2014.

[10] Yukihiro Matsumoto. Re: Ruby's lisp features. http://blade.nagaokaut.ac.jp/cgi-bin/scat.rb/ruby/ruby-talk/179642. Última visita el 17/12/2014.

[11] Ruby Community. Más allá de lo básico. viendo todo como un objeto. https://www.ruby-lang.org/es/about/. Última visita el 17/12/2014.

[12] Erik Trautman. Ruby explained: Objects and methods.

[13] Eric Lippert. What is "duck typing¿ http://ericlippert.com/2014/01/02/what-is-duck-typing/comment-page-2/. Última visita el 17/12/2014.

[14] David Heinemeier Hansson. Getting started with rails. http://guides.rubyonrails.org/getting_started.html. Última visita el 18/12/2014.

[15] Artículo Colaborativo. Convention over configuration. http://http://en.wikibooks.org/wiki/Ruby_on_Rails/Getting_Started/Convention_Over_Configuration. Última visita el 18/12/2014.

[16] David Heinemeier Hansson. Active record basics. http://guides.rubyonrails.org/active_record_basics.html. Última visita el 18/12/2014.

[17] Harlow Ward. End-to-end testing with rspec integration tests and capybara. http://robots.thoughtbot.com/rspec-integration-tests-with-capybara. Última visita el 19/12/2014.

[18] Reuven M. Lerner. At the forge - cucumber. http://www.linuxjournal.com/magazine/forge-cucumber. Última visita el 19/12/2014.

[19] Heroku.inc. Heroku add-ons. add powerful functionality to your apps with ease. https://addons.heroku.com. Última visita el 20/12/2014.

[20] Dan Podsedly. Velocity matters. http://www.pivotaltracker.com/community/tracker-blog/velocity-matters. Última visita el 20/12/2014.

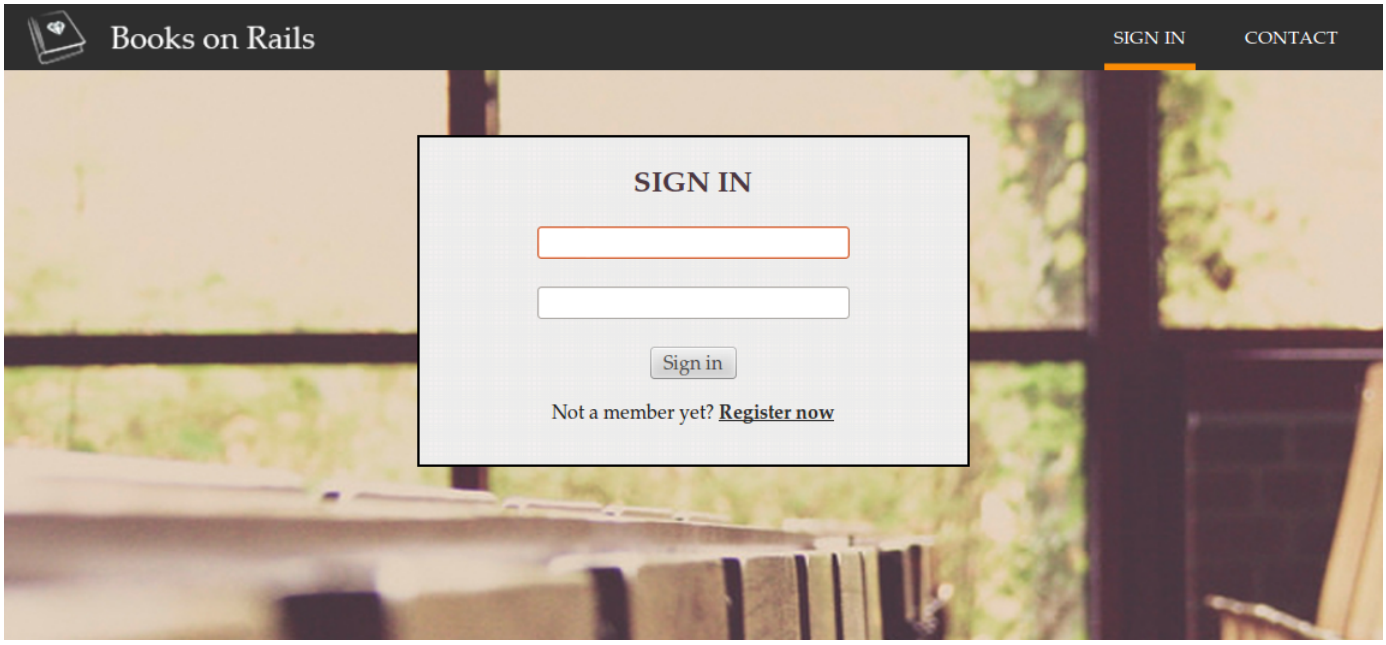

Figura 45: Página de acceso a la aplicación.

Apéndices

A Manual de usuario

Este anexo corresponde con el manual de uso de la aplicación web. Ilustra mediante capturas y explicaciones escritas cómo un usuario puede hacer un uso total de la aplicación.

A.1. Introducción

"Books on Rails" es una aplicación web accesible desde cualquier navegador. Con ella cualquiera puede planificar reuniones de intercambio de libros físicos y gestionar su propia biblioteca online para llevar un registro de los ejemplares que se tienen en casa.

Es una aplicación colaborativa en la que todo el mundo puede a la vez aportar y beneficiarse de la información de los demás.

A.2. Acceso a la aplicación

Para acceder a la web tenemos que meternos en `booksonrails.herokuapp.com` desde un navegador de cualquier ordenador de escritorio.

Lo primero que nos encontramos es la página de acceso a la web (ver Figura 45). Este acceso se realiza con una cuenta que hemos creado previamente.

En la barra del menú superior podemos observar los distintos apartados de la aplicación. La muesca de color naranja bajo el nombre indica en qué apartado nos encontramos actualmente.

Podemos acceder a la página de información de contacto (ver Figura 46). En esta se explica para qué sirve Books on Rails, por qué se ha realizado este proyecto, y la información de contacto tanto del estudiante desarrollador de la aplicación como de la profesora supervisora del mismo.

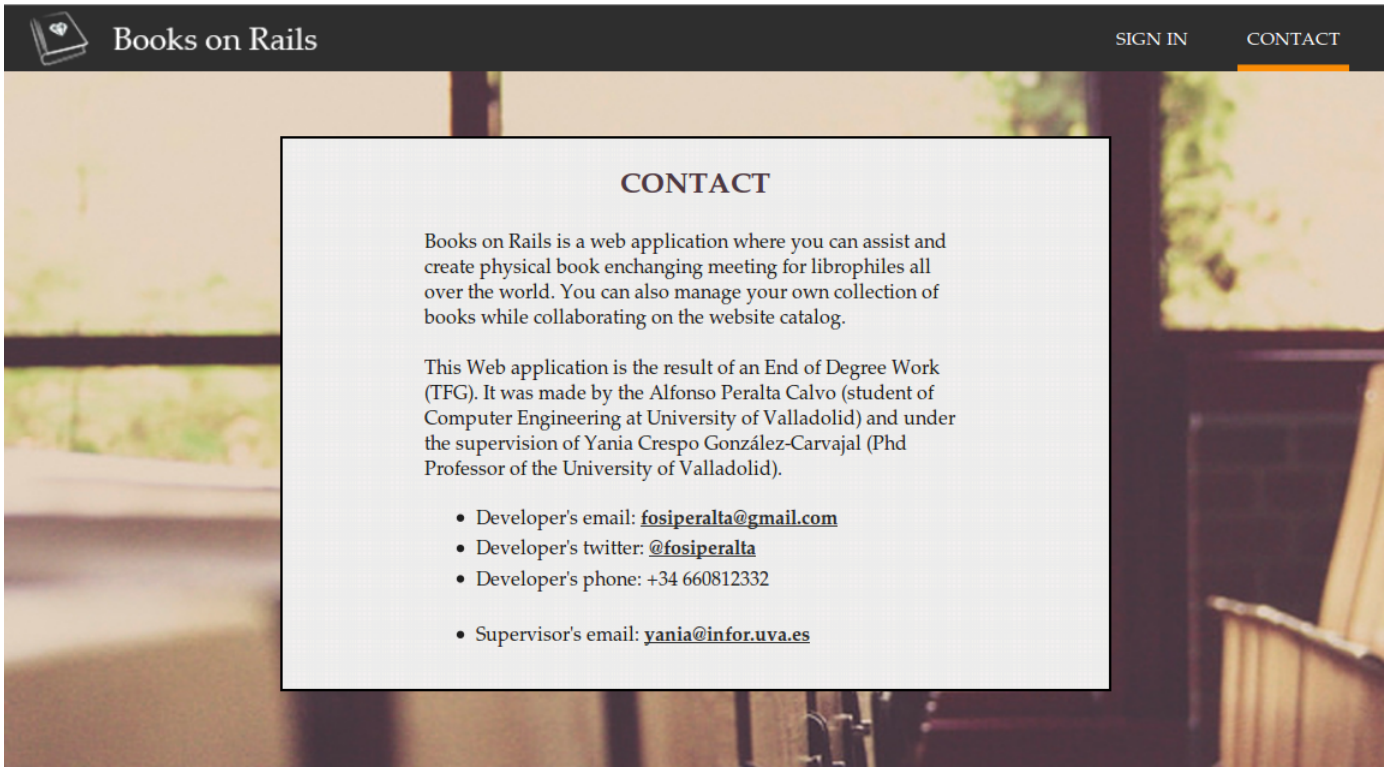

Figura 46: Información de contacto.

A.3. Creando una nueva cuenta de usuario

Para acceder al sistema necesitamos una cuenta de usuario, desde la página de inicio de sesión tenemos un enlace a la página de registro de usuarios (ver Figura 47). Introducimos los datos necesarios y le damos a registrar.

A.4. Ingresando en el sistema

Si todo ha ido bien en el registro, introduciremos los datos en la página de acceso (ver Figura 48). e ingresaremos en la aplicación exitosamente.

La página que nos encontraremos al acceder es la página de bienvenida (ver Figura 49). En ella se muestra información de la aplicación y de los diferentes apartados y funcionalidades que la aplicación ofrece.

La barra de menú superior ha cambiado de manera que ahora vemos también los principales apartados o funcionalidades de la aplicación.

A.5. Gestionando el catálogo de libros

La primera gran funcionalidad que vamos a explicar es el catálogo de libros. Este catálogo es una lista de los libros almacenados en toda la aplicación. Tiene un método de gestión colaborativa de modo que todo el mundo puede añadir nuevos libros, editar los existentes o borrarlos.

Vemos una lista de libros (ver Figura 50) con cierta información sobre ellos (título, autor, idioma...). En caso de que queramos ver más información sobre un libro en concreto, no tenemos más que hacer click en su nombre.

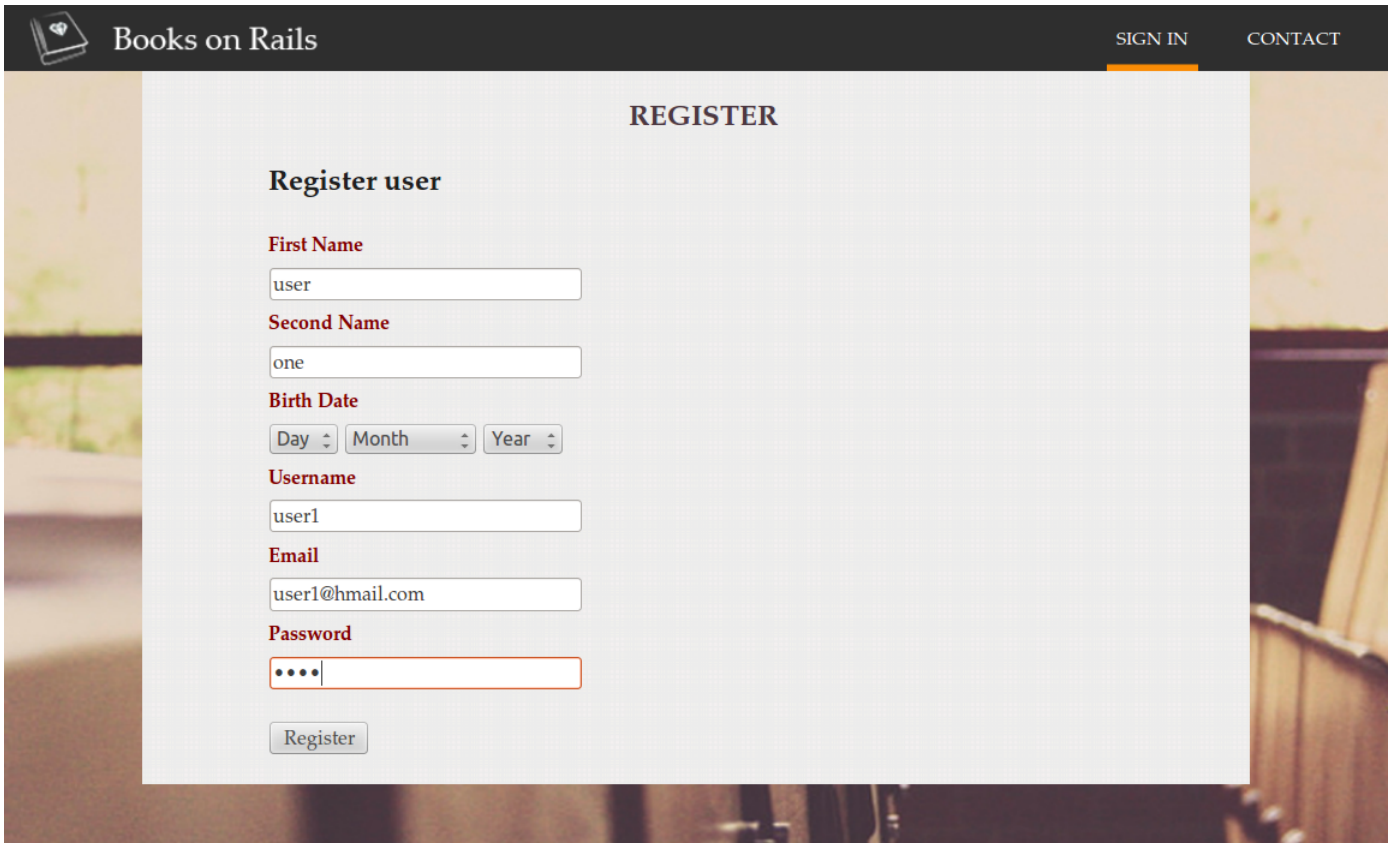

Figura 47: Creación de una nueva cuenta de usuario.

Figura 48: Acceso a la aplicación con una cuenta ya creada.

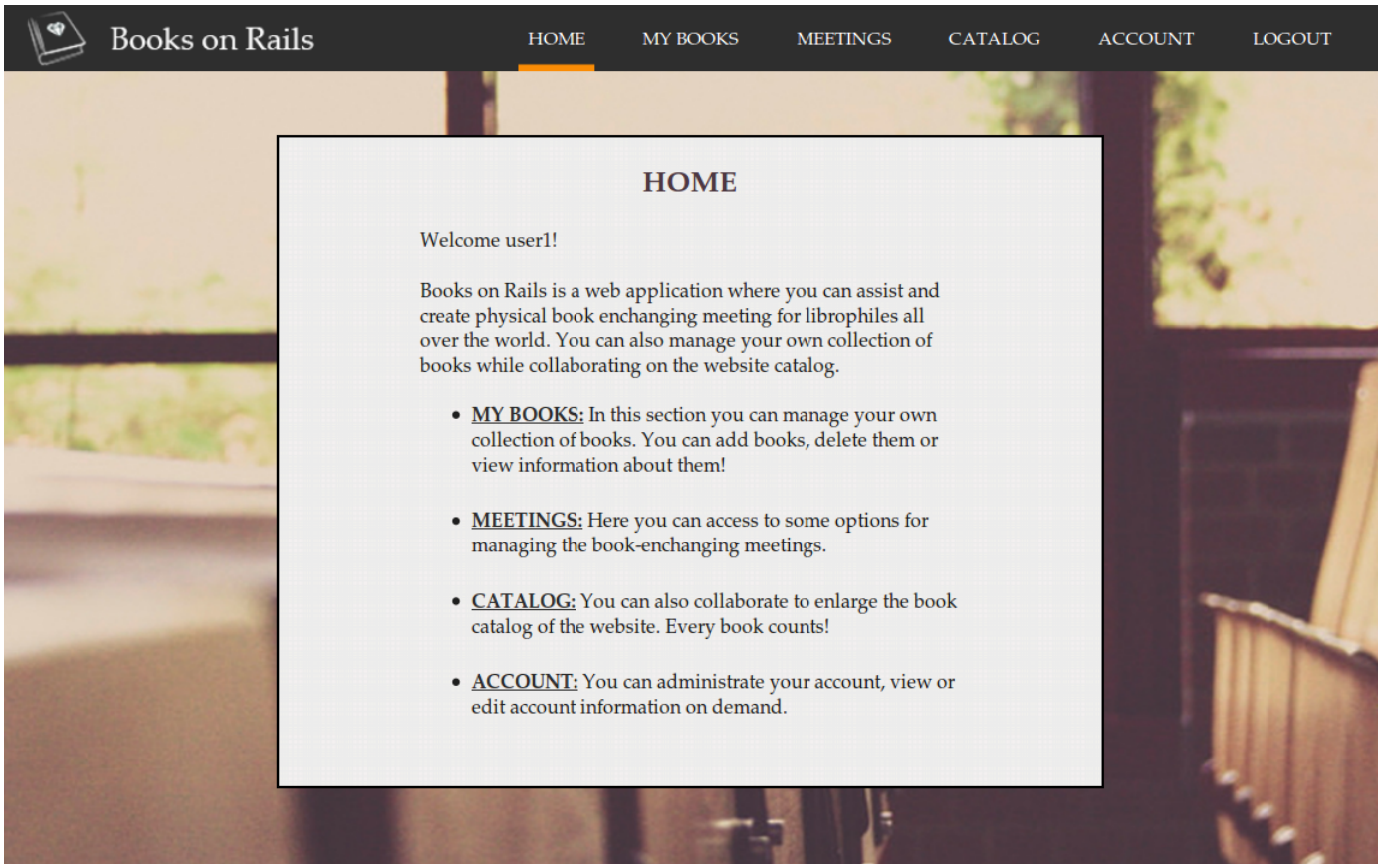

Figura 49: Página de bienvenida o Home.

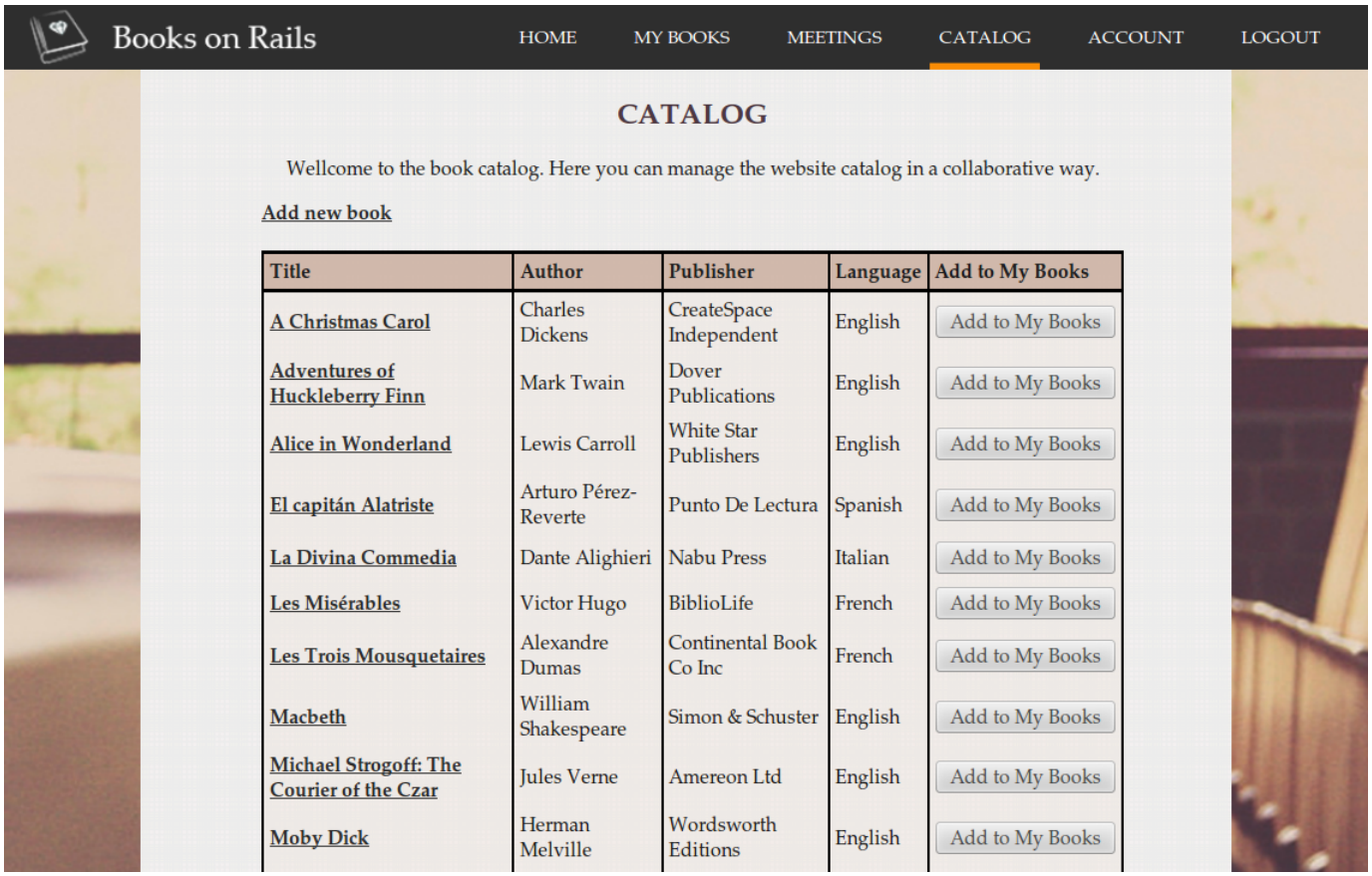

Figura 50: Catálogo de libros.

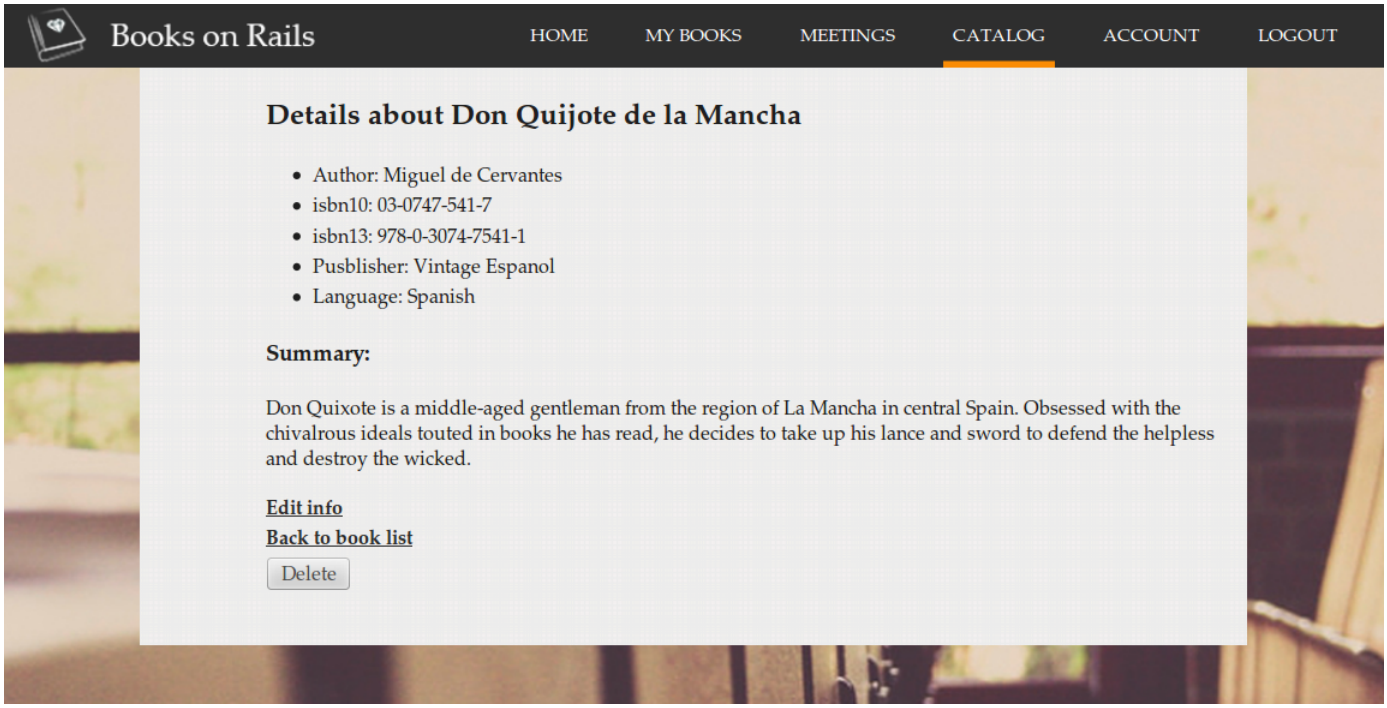

Figura 51: Información de un libro concreto.

Al hacer click en el nombre de un libro se nos abrirá una página en la que vemos información ampliada (códigos isbn10 e isbn13, un resúmen...) sobre ese libro (ver Figura 51).

A.5.1. Añadiendo un nuevo libro

Si se desea añadir un nuevo libro al catálogo general, en la página de la lista del catálogo hay un enlace llamado "Add new book". Al hacer click en él se nos abrirá una página con un formulario que rellenar con la información del libro (ver Figura 52).

Hay que prestar especial atención en introducir códigos ISBN10 e ISBN13 válidos tanto en el formato como en el dígito de control de cada uno.

Al añadir un nuevo libro, la aplicación nos avisará con un mensaje con texto sobre fondo verde (ver Figura 53).

A.5.2. Editando un libro

En la página de información del libro, tenemos un enlace que nos permite editar la información de un libro en caso de que esta sea errónea o deba ser modificada (ver Figura 54). La página es similar a la de añadir un nuevo libro, con un cuadro de texto para cada dato, y al hacer click en "Save Changes" se actualiza la información.

A.5.3. Borrando un libro

También podemos borrar un libro del catálogo, esta opción está accesible dentro de la página de información del libro en cuestión. Se nos abrirá una ventana emergente, la cual nos pedirá confirmación del borrado del libro.

Al borrar el libro, la aplicación nos notificará con un mensaje la acción que ha sucedido (ver Figura 55).

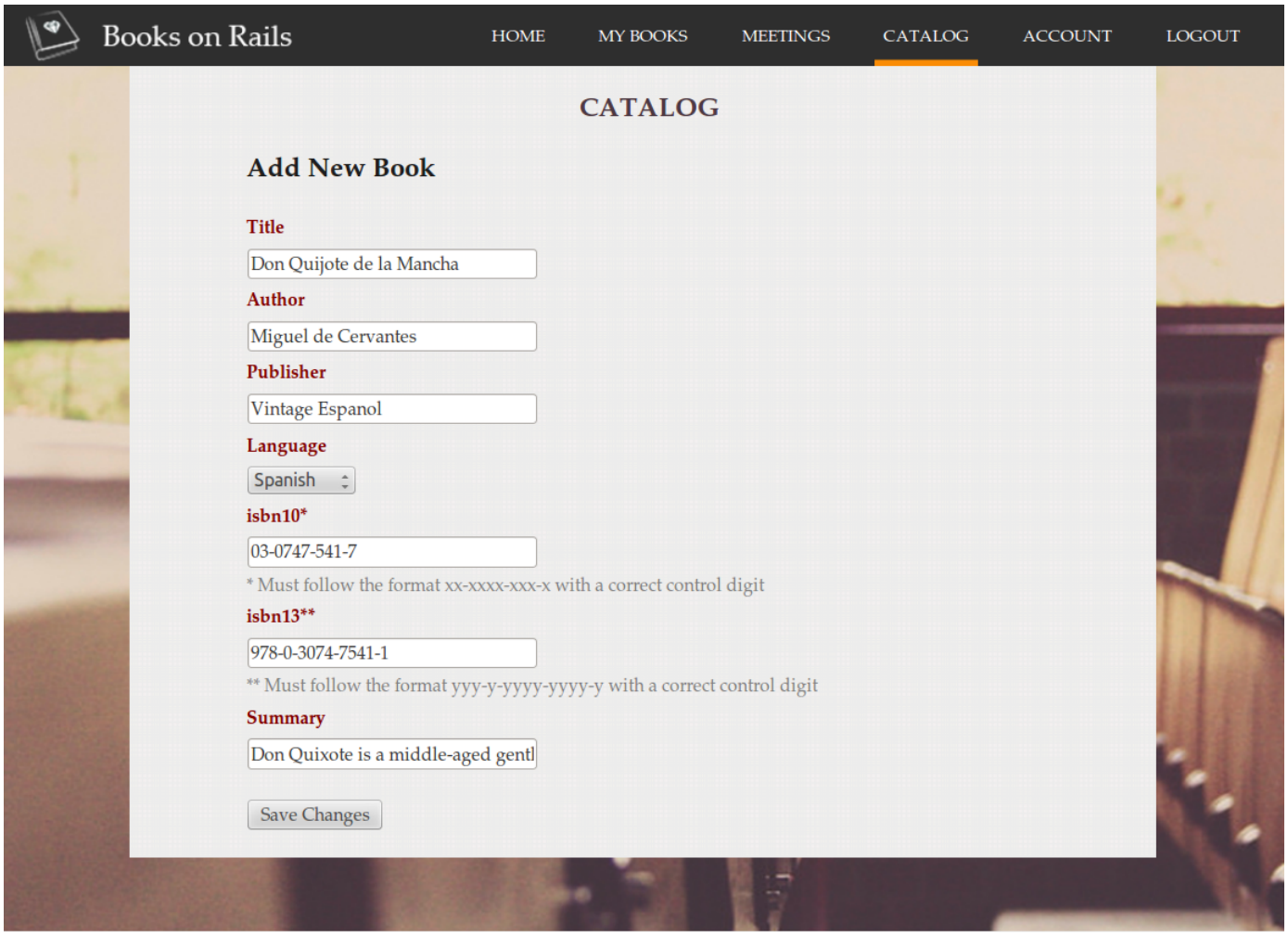

Figura 52: Adición de un nuevo libro.

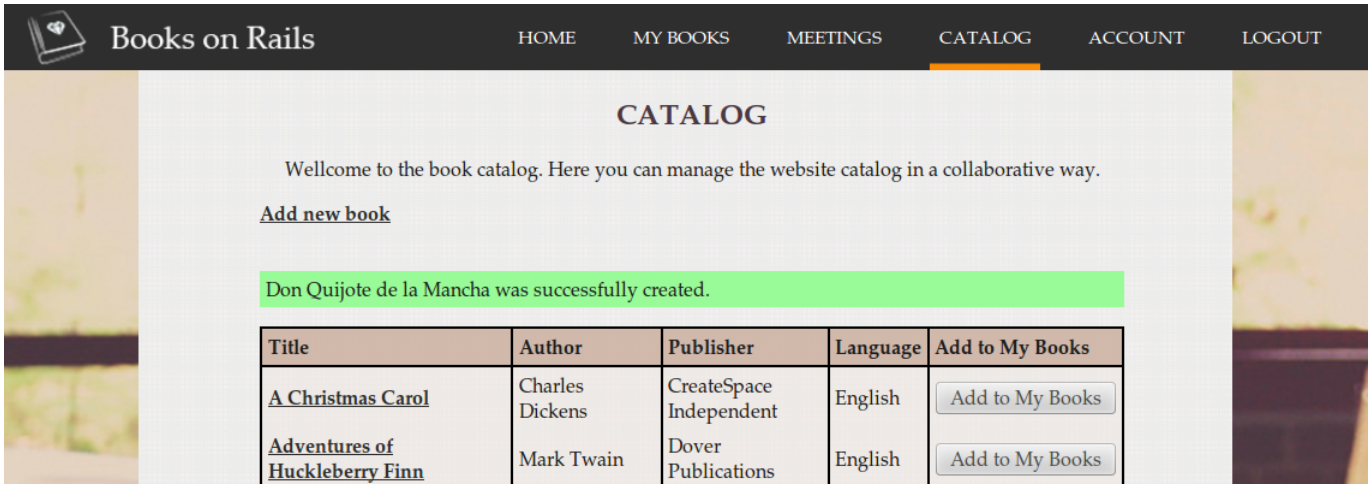

Figura 53: Mensaje superior de confirmación de creación.

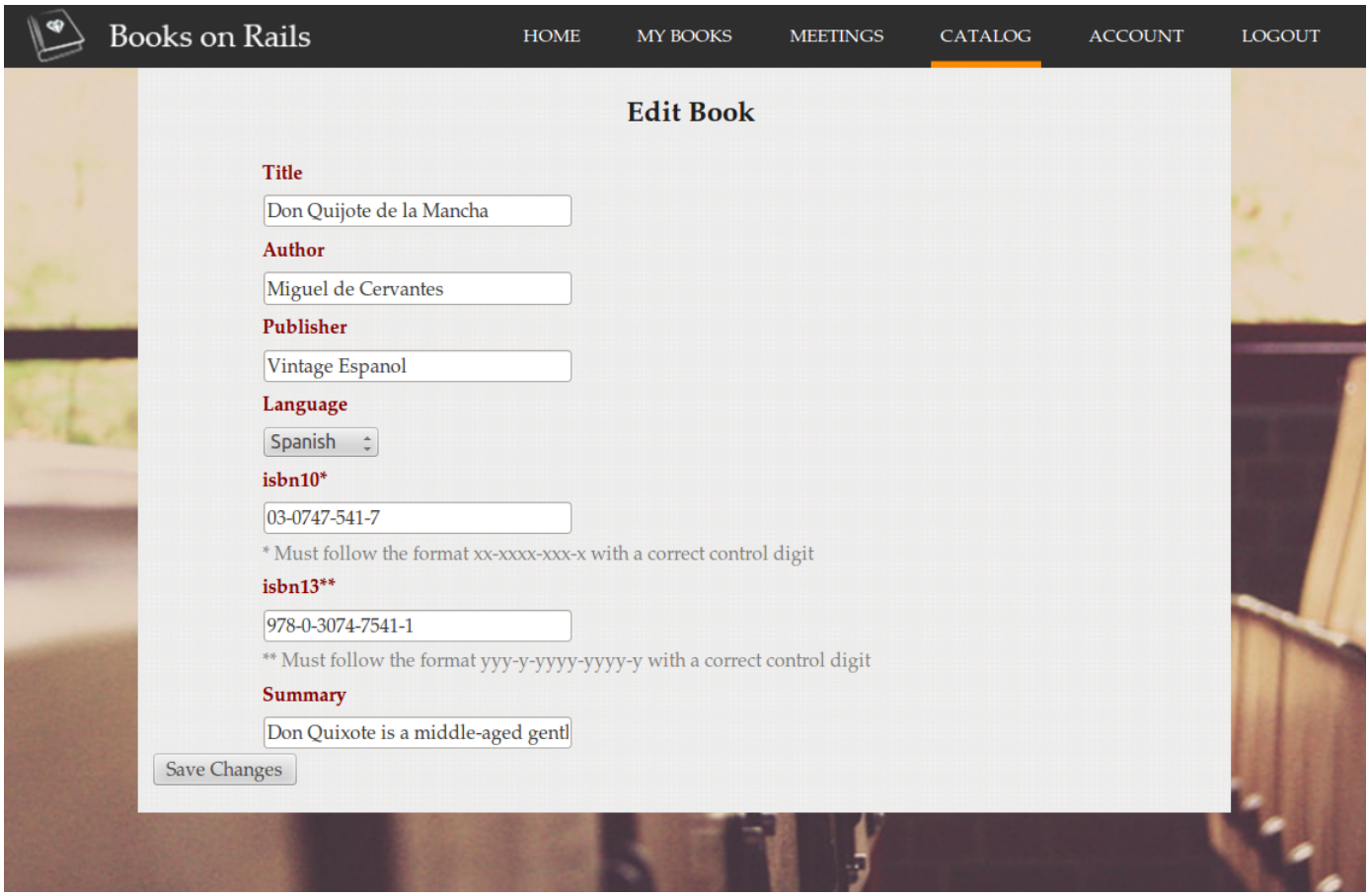

Figura 54: Edición de un libro existente.

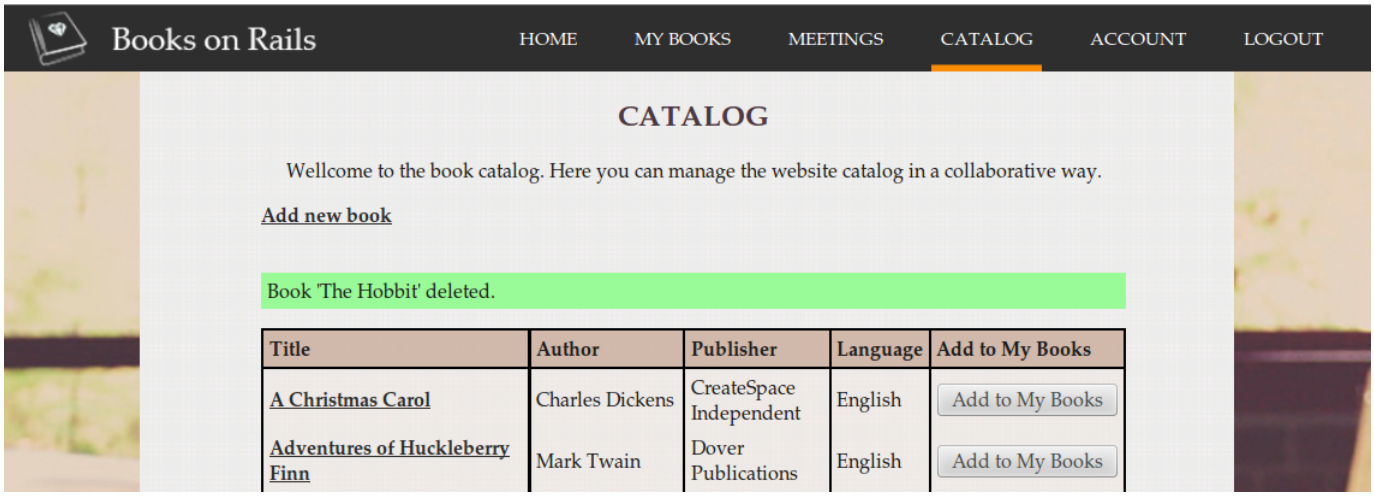

Figura 55: Mensaje superior de confirmación de borrado.

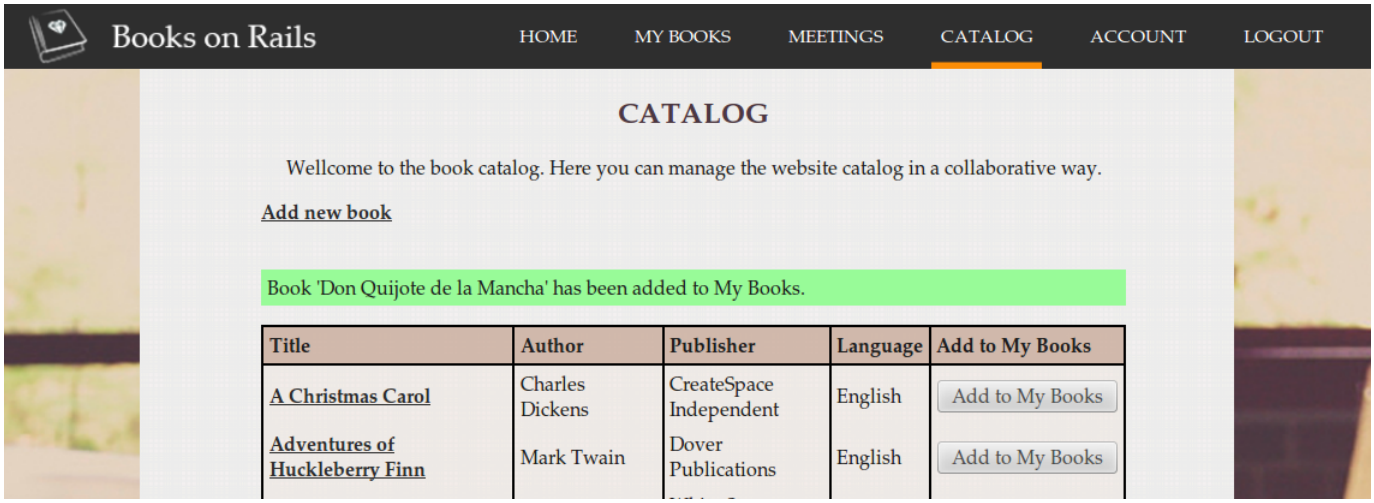

Figura 56: Adición un ejemplar de un libro a la colección personal.

A.6. Gestionando la colección personal de ejemplares

La segunda gran funcionalidad de la aplicación es la de poder gestionar nuestra propia colección de ejemplares, en la que tenemos un registro de todos los ejemplares de libros que poseemos físicamente.

A.6.1. Añadiendo un ejemplar

Desde la lista de libros del catálogo podemos hacer click en "Add to My Books" en cualquiera de ellos para añadirlos a nuestra colección particular.

Al añadir un ejemplar, el sistema nos notificará si el ejemplar se ha añadido (ver Figura 56) o si por el contrario ya poseíamos ese ejemplar en nuestra colección.

Si navegamos al apartado de My Books en el menú superior, veremos una lista con todos los ejemplares de libros que tenemos en nuestra colección (ver Figura 57), pudiendo también acceder a la información de cada uno de ellos.

A.6.2. Retirando un ejemplar

En esta lista también se ofrece la opción de retirar un ejemplar de la colección con el botón "Remove". Al hacerlo la aplicación nos notificará que efectivamente se ha retirado el ejemplar (ver Figura 58), y este habrá desaparecido de la lista.

A.6.3. Buscando y añadiendo ejemplares desde My Books

Desde nuestra colección particular podemos buscar cualquier libro cuyo título contenga cierta cadena de texto. Al apretar "Search" aparecerá una lista de libros que se adecúen a esos términos de búsqueda (ver Figura 59).

Desde esta lista podemos añadirlos rápidamente como ejemplares a nuestra colección pulsando "Add to My Books" y de nuevo recibiremos una notificación confirmando que se ha añadido correctamente (ver Figura 60).

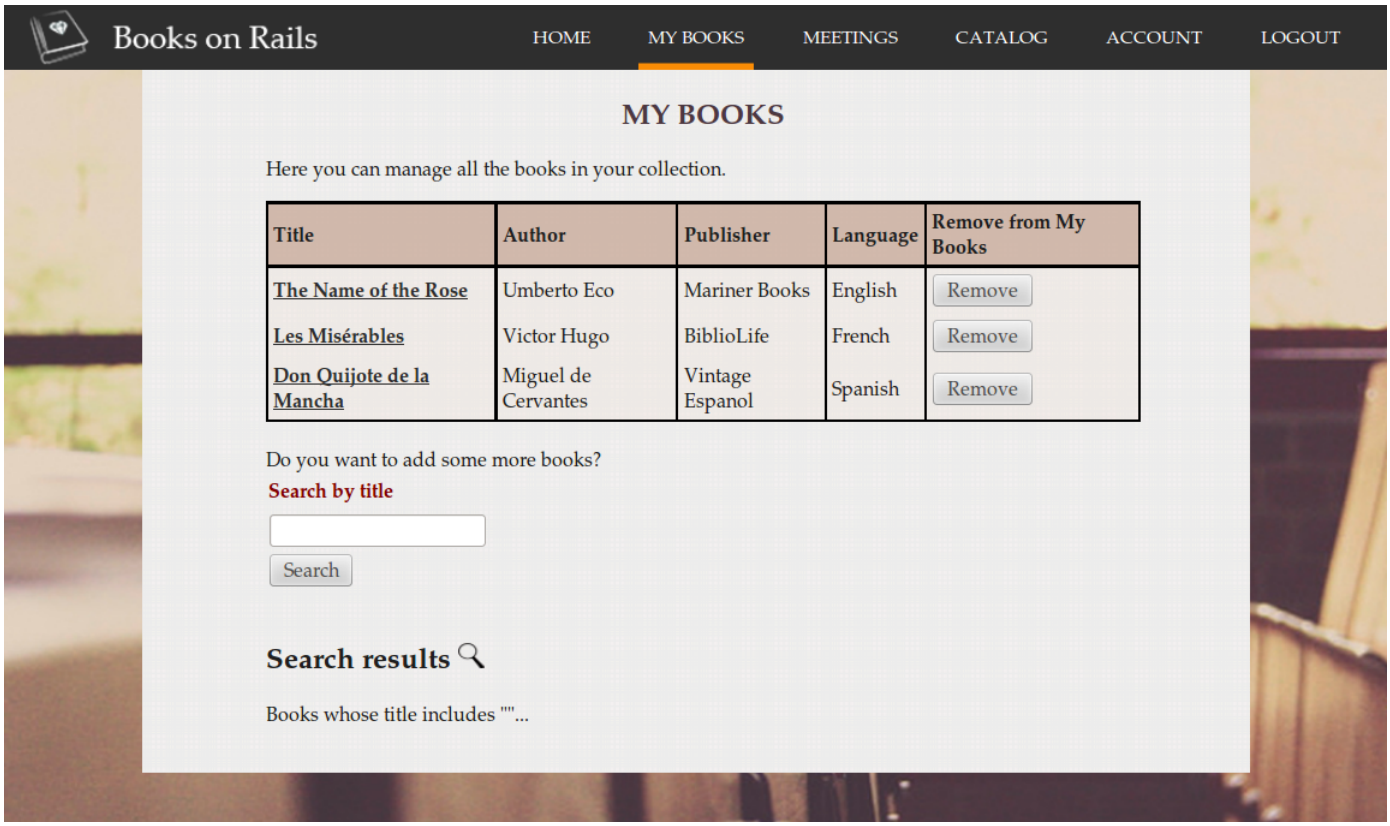

Figura 57: Colección personal de ejemplares.

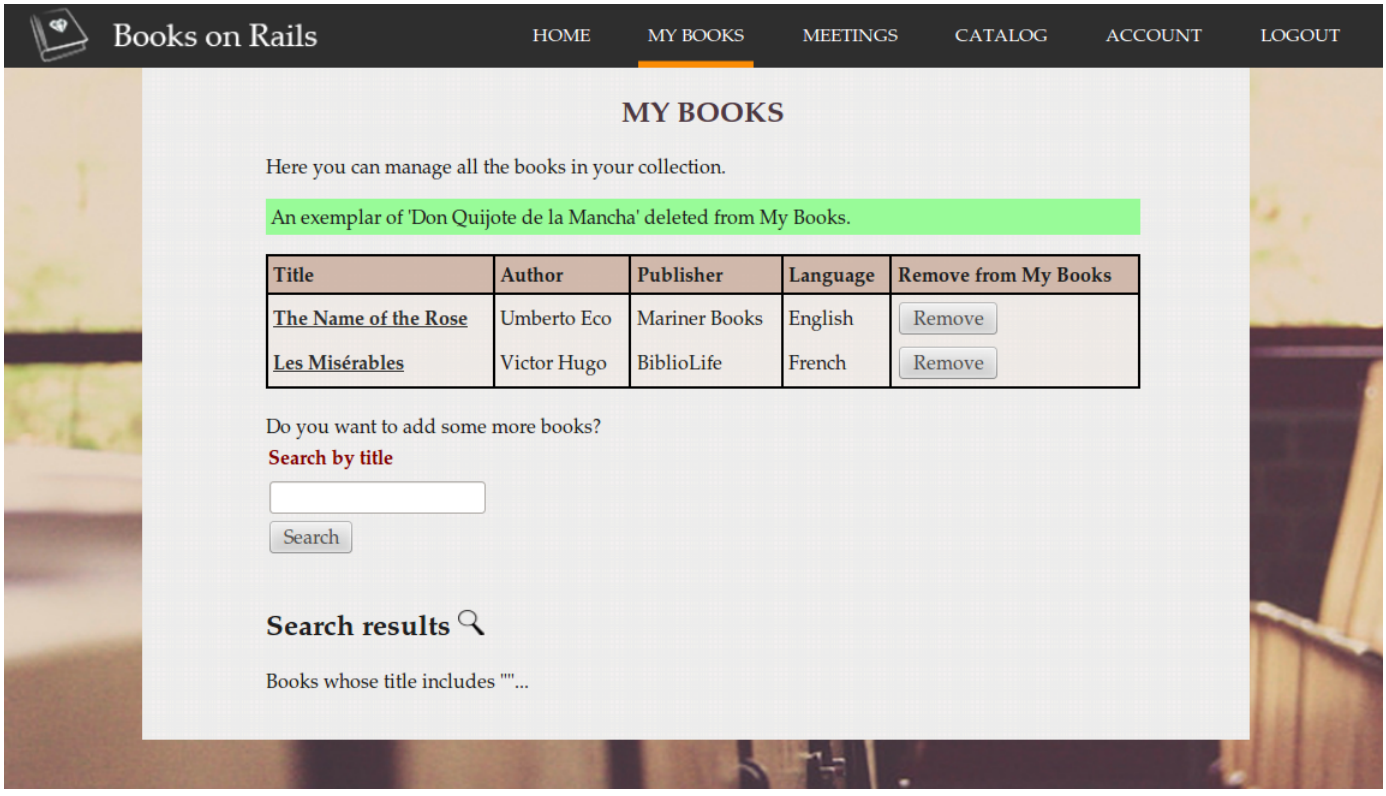

Figura 58: Mensaje superior de confirmación de borrado.

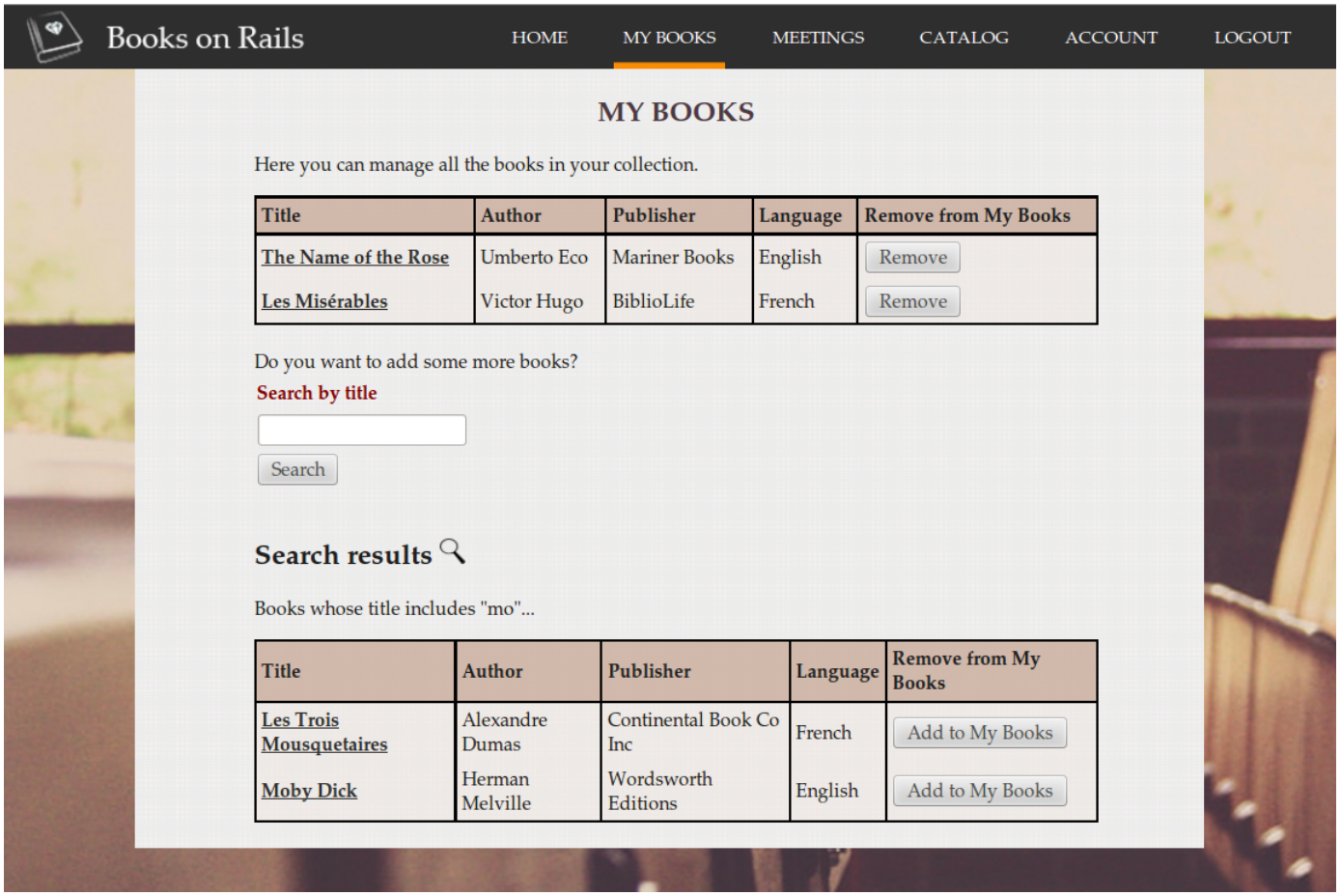

Figura 59: Buscar desde “My Books”.

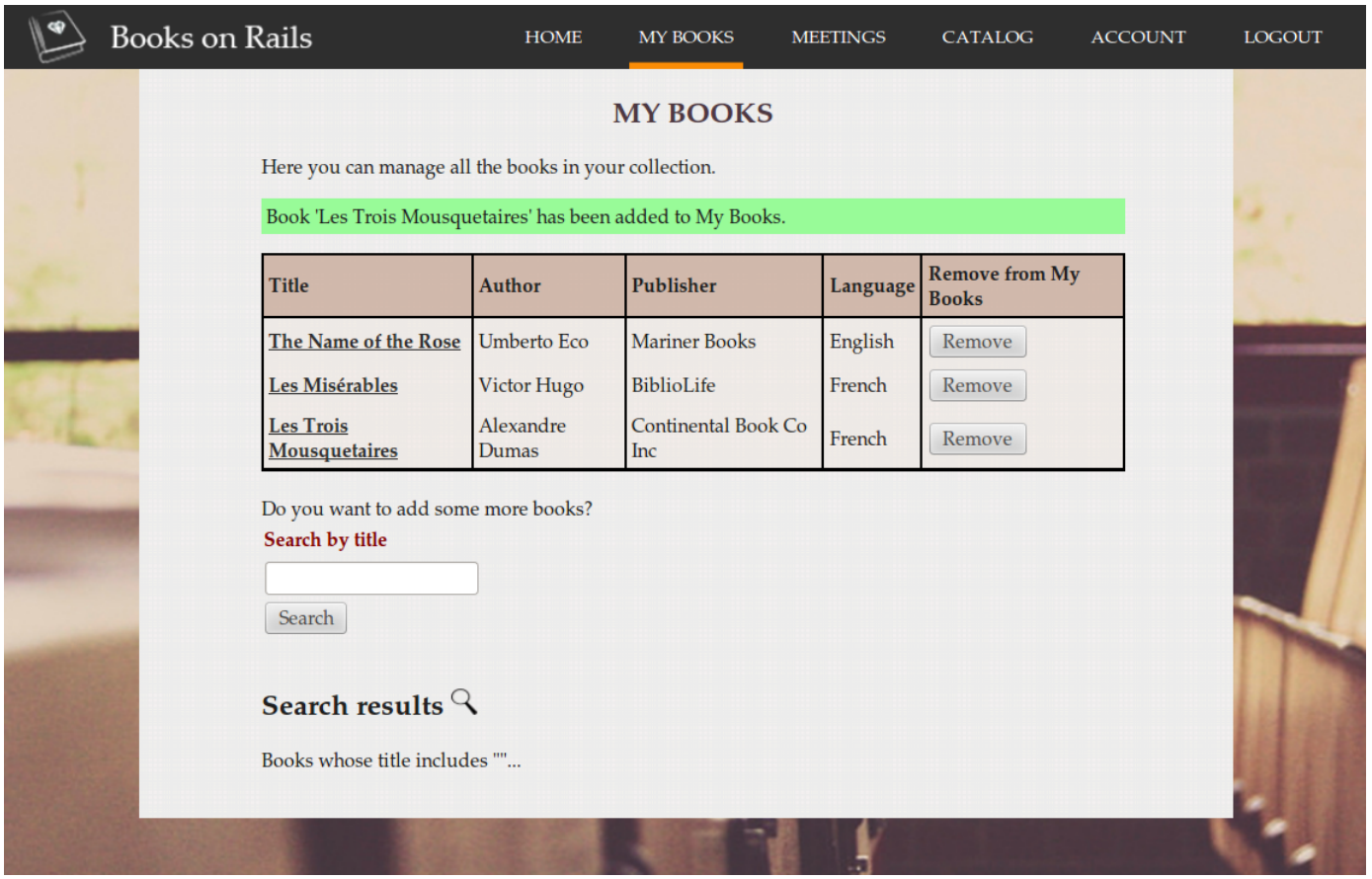

Figura 60: Añadir despueés de buscar desde “My Books”.

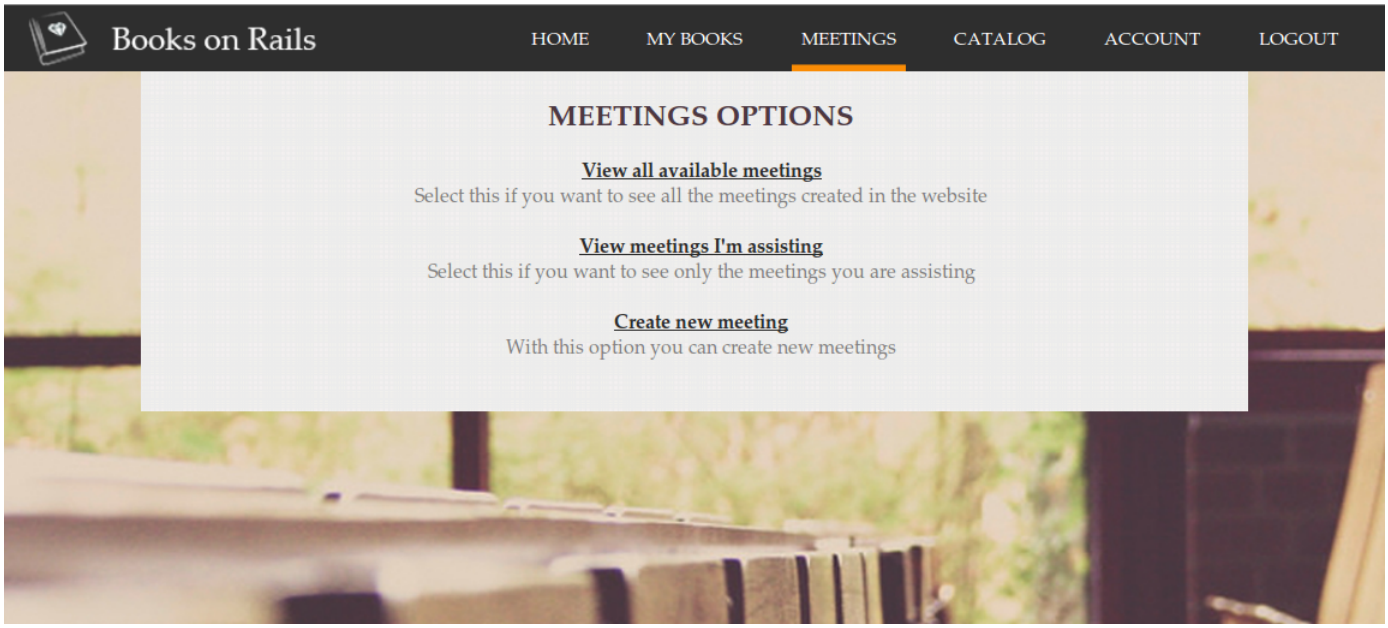

Figura 61: Opciones del apartado de Meetings.

A.7. Gestionando los meetings

La siguiente gran funcionalidad es la posibilidad de gestionar reuniones de intercambio de libros físicos. A estas reuniones la gente confirma su asistencia e indica qué libros físicos va a llevar para intercambiar con otros usuarios.

Si seguimos el enlace de la barra de menú superior, accedemos a una página con las distintas opciones de gestión de reuniones (ver Figura 61).

A.7.1. Creando un nuevo meeting

Si seleccionamos la opción de "Create new meeting" accedemos a una página en la que introducimos la información de un nuevo meeting que vamos a crear (ver Figura 62), y al hacer click en "Save Changes" la reunión se crea y almacena en la aplicación.

Si la reunión se ha creado correctamente, el sistema nos lo notifica con un aviso de texto en fondo verde (ver Figura 63). En esta página podemos ver los meetings que han sido creados en el sistema.

A.7.2. Visualizando un meeting

Si hacemos click en el nombre de la reunión en la lista de reuniones, podemos visualizar toda la información relativa a ese meeting (ver Figura 64). Esta incluye qué usuarios van a asistir a ese meeting y qué libros van a llevar esos usuarios.

A.7.3. Editando un meeting

Desde la página de información de meeting (reunión) tenemos la opción de editar información de ese meeting si somos nosotros los creadores del mismo (ver Figura 65).

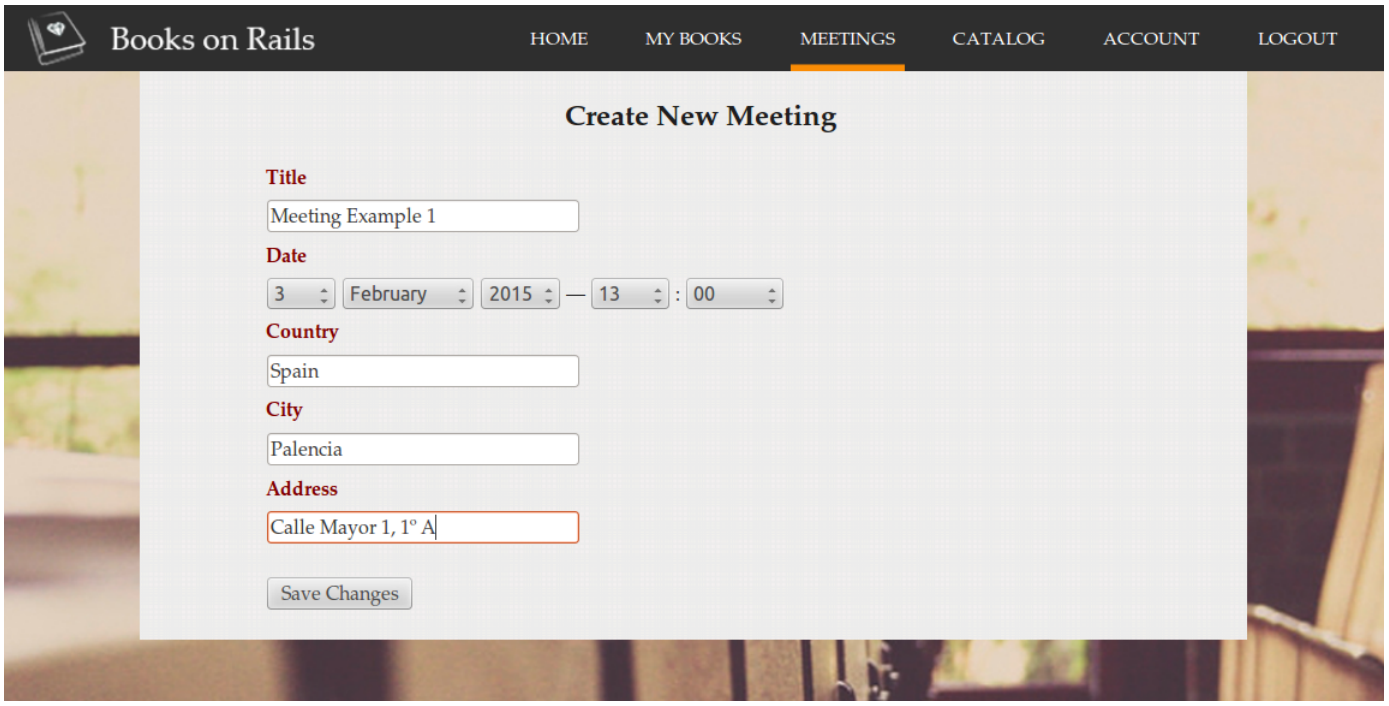

Figura 62: Creación de un nuevo meeting.

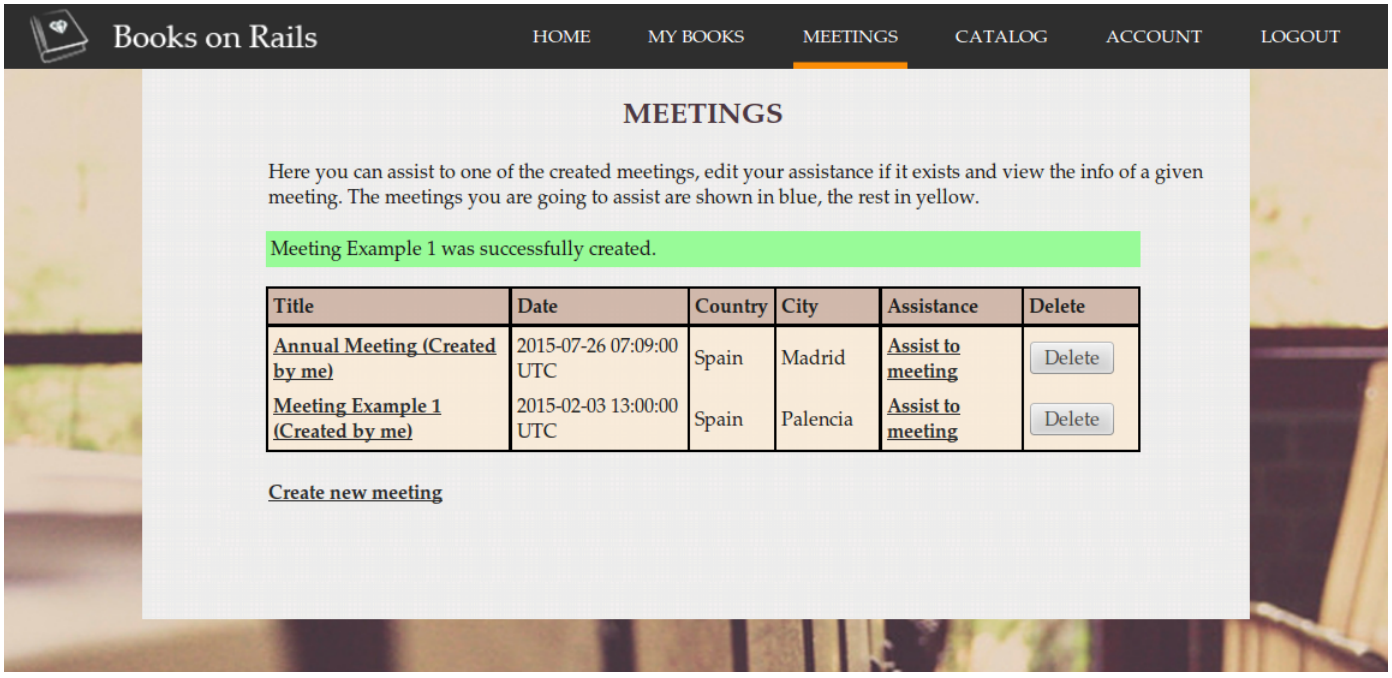

Figura 63: Mensaje superior de confirmación de creación.

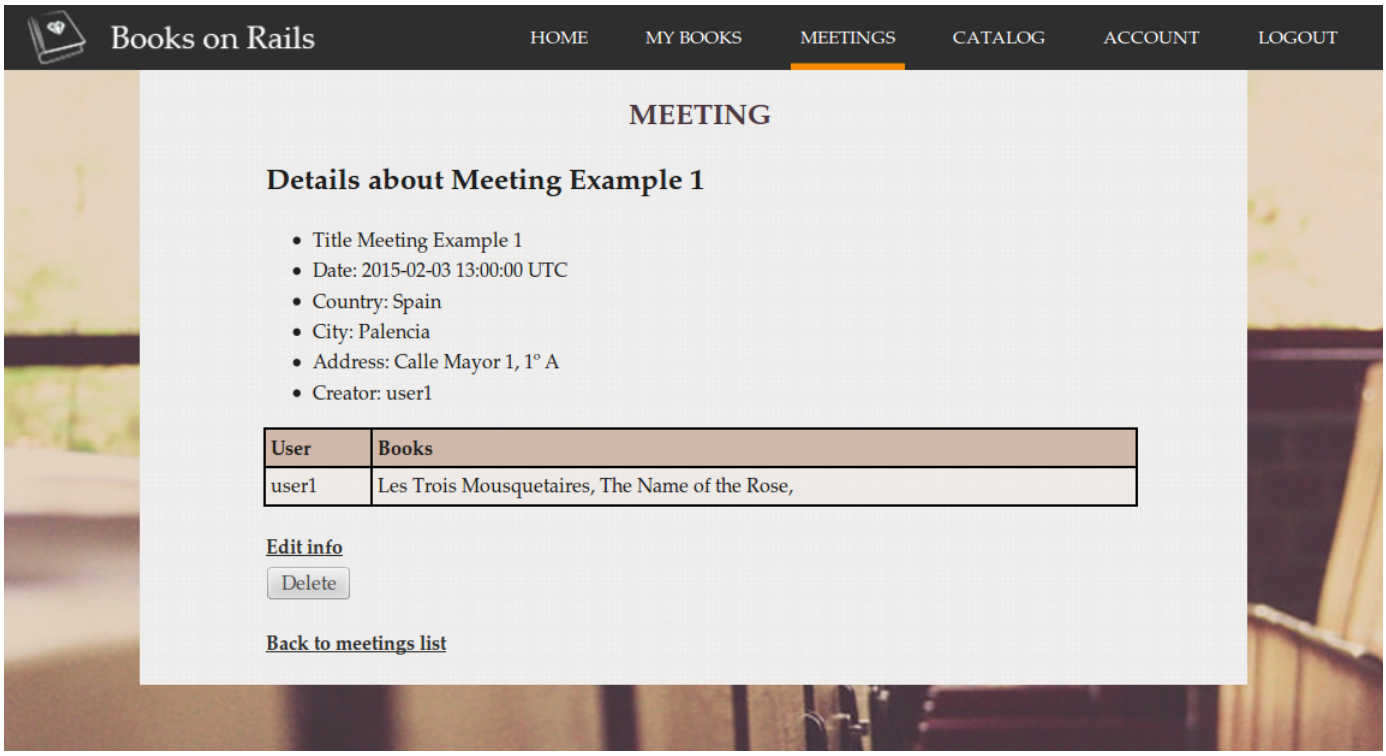

Figura 64: Información sobre un meeting concreto.

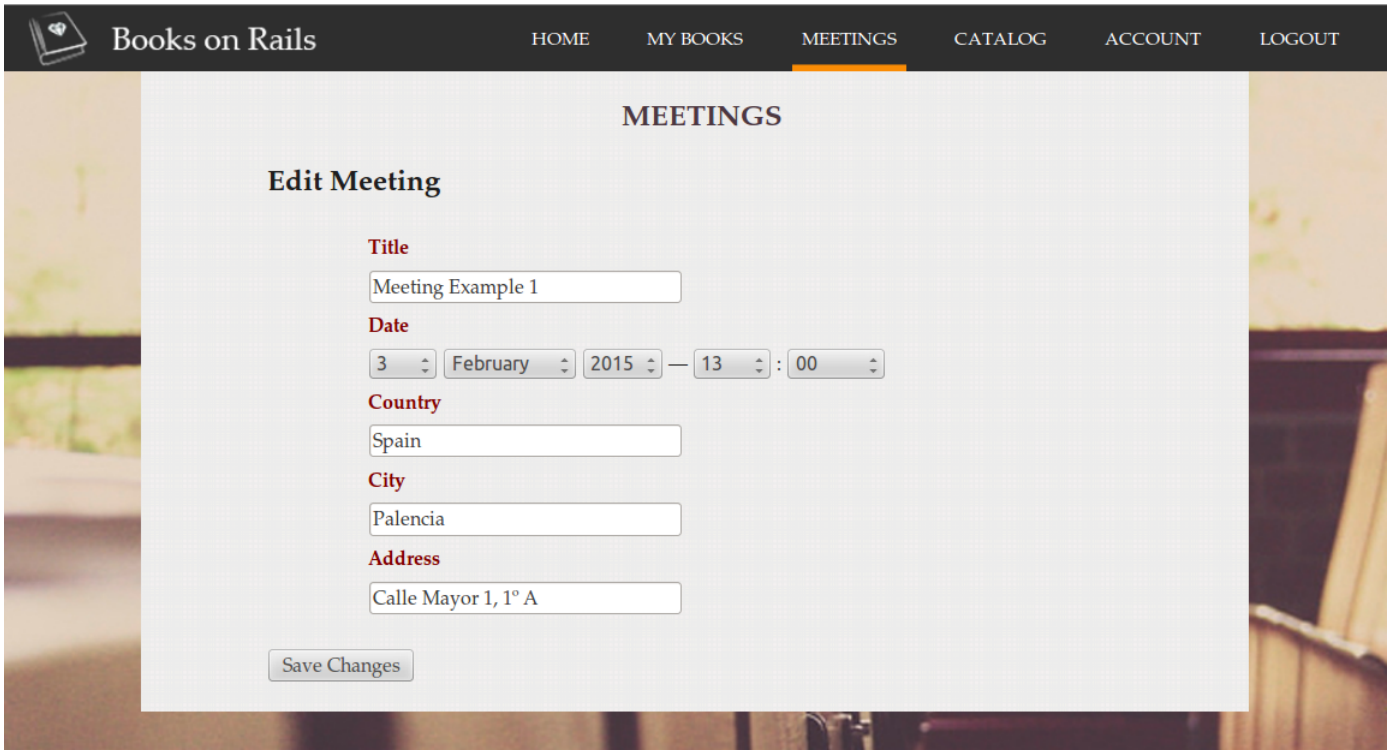

Figura 65: Edición de un meeting existente.

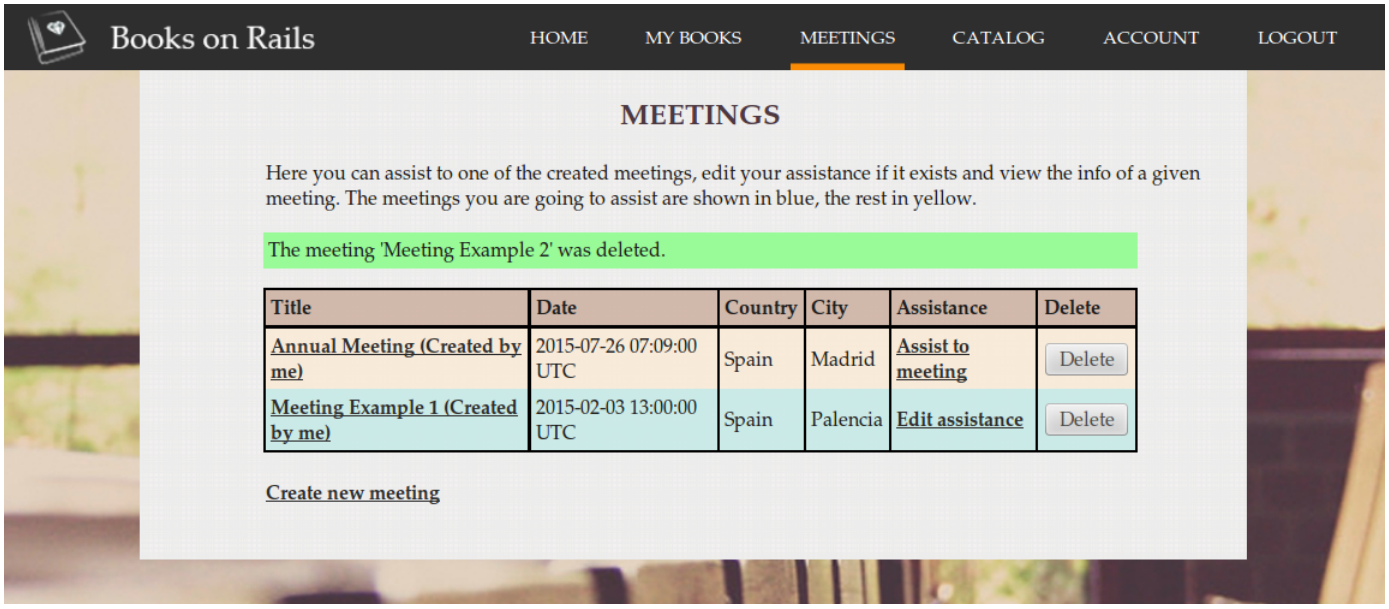

Figura 66: Mensaje superior de confirmación de borrado.

A.7.4. Borrando un meeting

Tanto desde la lista de meetings, como desde la de información de uno en concreto, tenemos la opción de eliminarlo, solamente si somos nosotros los creadores de ese meeting. Al hacer click en "Delete" la aplicación nos mostrará un mensaje confirmándolo (ver Figura 66).

A.8. Gestionando las asistencias a meetings

Hemos visto como crear un meetingo editar uno existente, aquí se explica cómo apuntarse a uno ya creado y decir qué ejemplares de libros vamos a llevar a ese meeting.

A.8.1. Asistiendo a un meeting

Para notificar al sistema que vamos a asistir a un meeting, en la lista de meetings hacemos click en "Assist to meeting" de la fila del meeting que deseemos. Entonces se nos abrirá una página con información del meeting y la lista de ejemplares de libros que tenemos en nuestra colección (ver Figura 67).

Para agregar uno de esos ejemplares al meeting (decirle a la aplicación que vamos a llevar ese ejemplar) hacemos click en "Take book to meeting". Entonces el sistema nos notificará que ese ejemplar se va a llevar a ese meeting y la fila del ejemplar en cuestión se volverá azul (ver Figura 68), los que no vayamos a llevar seguirán en color amarillo.

Al haber indicado nuestra asistencia a un meeting a la aplicación, esta lo sabrá y a partir de ese momento mostrará la lista con la fila del meeting en azul (ver Figura 69).

Si lo deseamos, podemos ver una lista con los meeting a los que hayamos indicado nuestra asistencia y podremos visualizar qué libros vamos a llevar (ver Figura 70). Desde aquí podemos retirar nuestra asistencia a ese meeting.

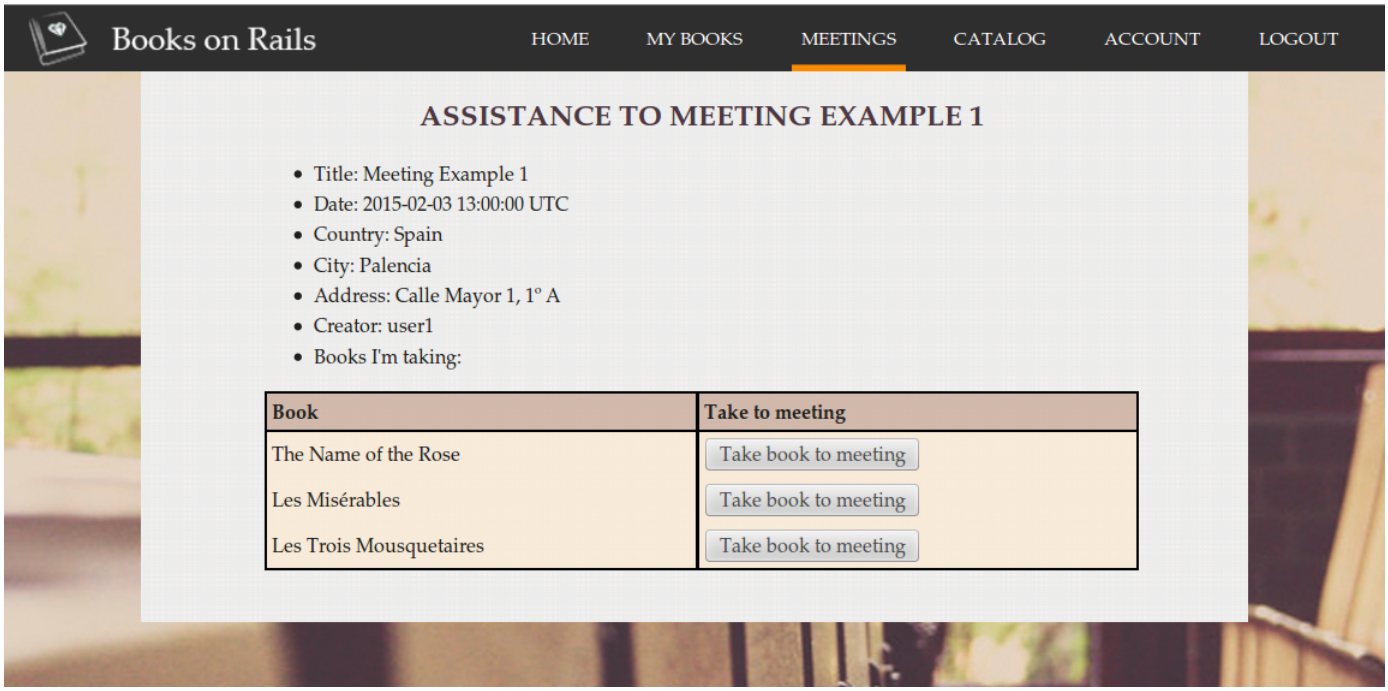

Figura 67: Creación una asistencia a un meeting concreto.

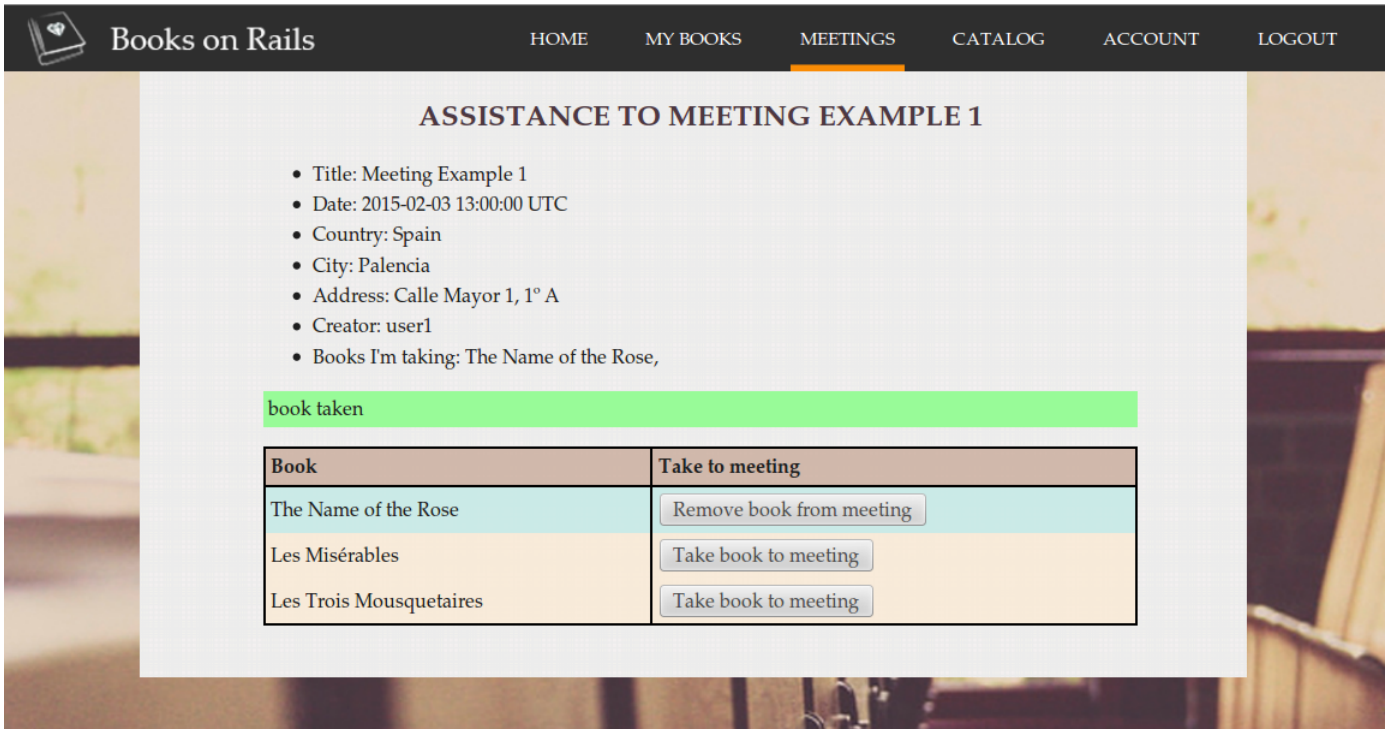

Figura 68: Marcado de un libro para llevar al meeting.

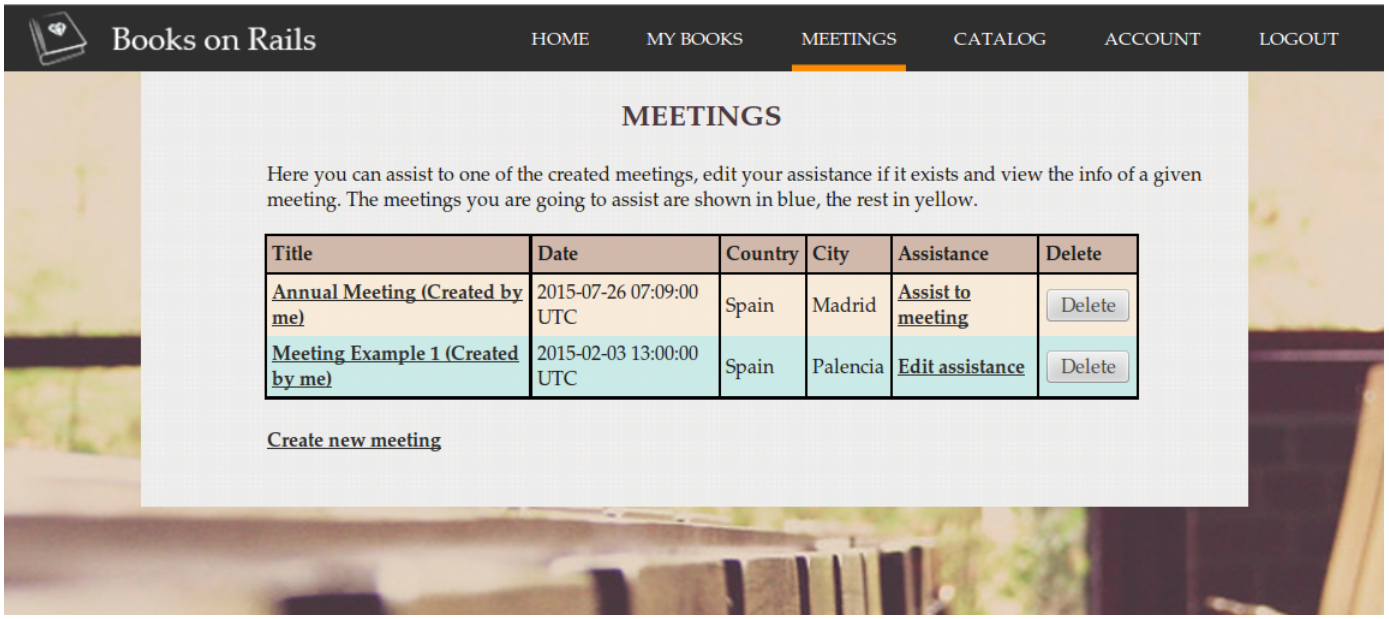

Figura 69: Lista de meetings existentes.

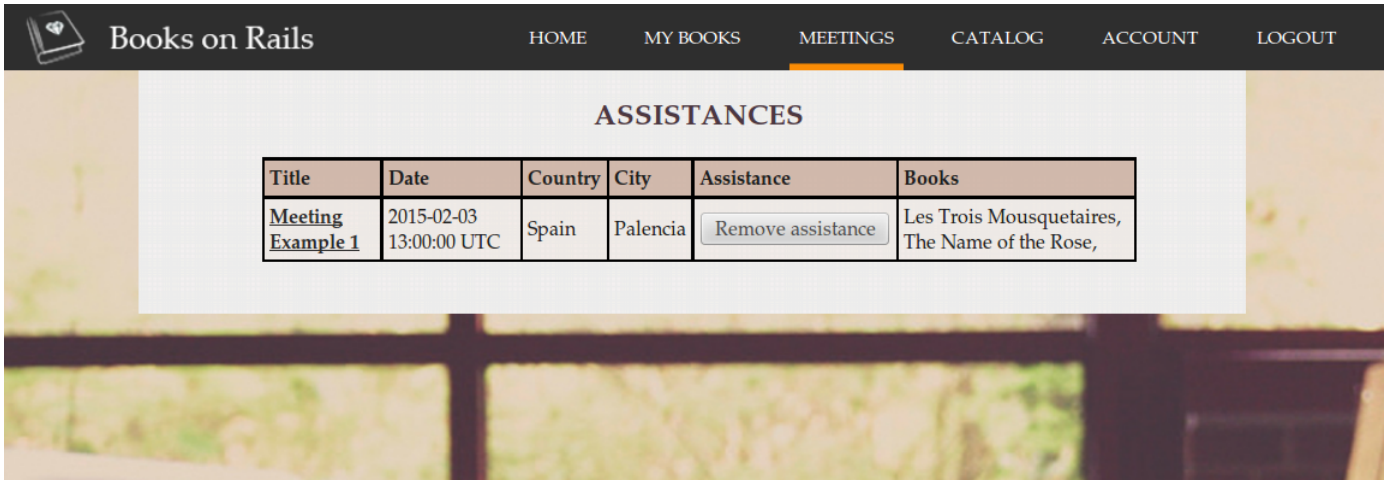

Figura 70: Lista de meetings a los que se va a asistir.

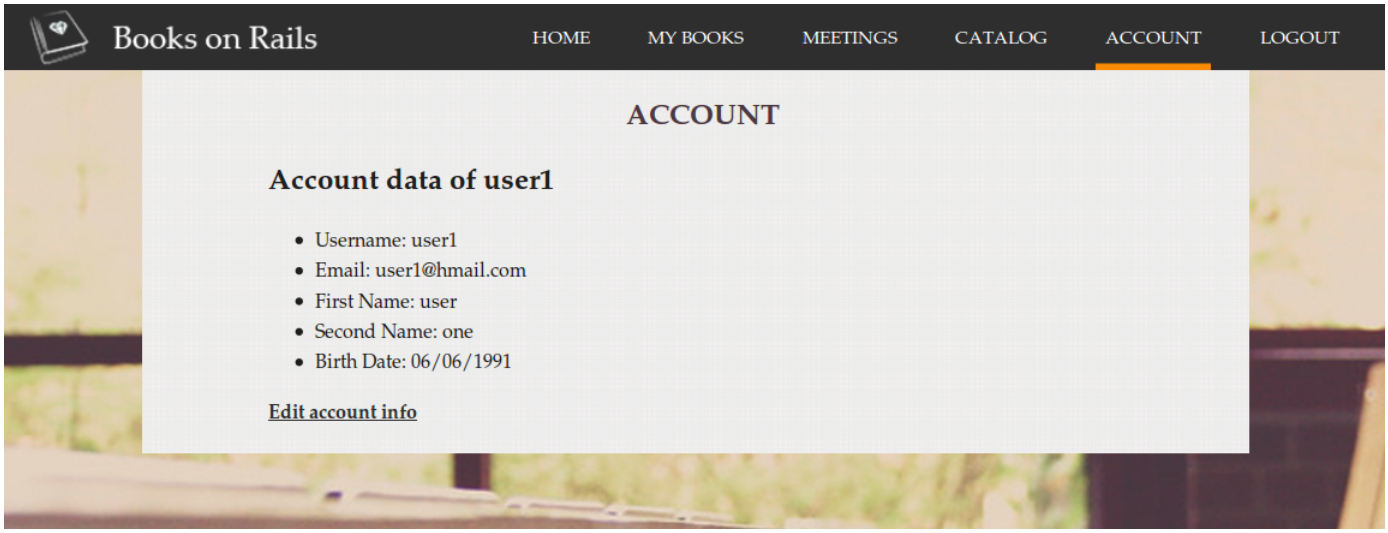

Figura 71: Información de la cuenta del usuario.

A.9. Gestionando la información de cuenta de usuario

Si lo deseamos podemos visualizar la información de nuestra cuenta de usuario en el apartado "ACCOUNT" en el menú superior. Veremos toda la información que introdujimos al crear nuestra cuenta de usuario (ver Figura 71).

A.9.1. Editando la información de usuario

Además podemos editar la información de nuestra cuenta si queremos modificar alguno de sus datos (ver Figura 72).

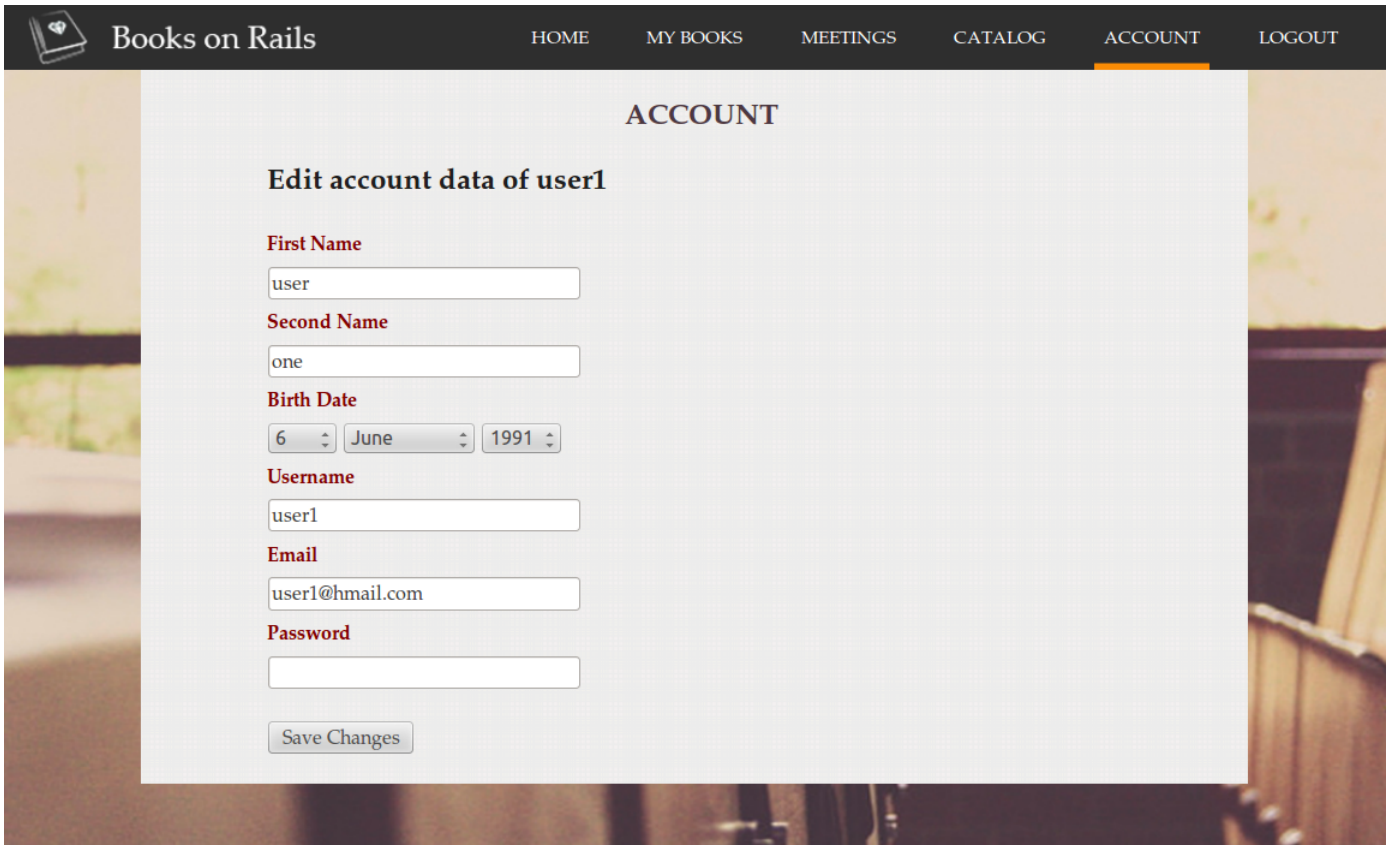

Figura 72: Edición de la información de cuenta de usuario.

B Report de Pivotal Tracker

B.1. Comentario y valoración del report

Pivotal Tracker (ver 3.7) nos ha ayudado a llevar un seguimiento del proyecto y a tratar de estimar la velocidad de producción de features en este desarrollo software.

Dado que era una herramienta que no se conocía, su uso no ha sido todo lo correcto que debiera y los resultados no reflejan en algunos momentos la velocidad de producción.

Si observamos el report, el día 24 de Noviembre se desarrollaron gran cantidad de features. Esto no es realmente así, si no que estas ya habían sido desarrolladas pero no introducidas en Pivotal Tracker. Así los resultados muestran un pico en la velocidad de producción inicial y un brusco descenso posterior. Estos datos no son por tanto representativos de la velocidad de producción obtenida.

B.2. Contenido del report

Pivotal Tracker ofrece la opción de exportar como texto los hitos relevantes en el desarrollo de features, chores y otro tipo de historias de usuario. A continuación mostramos la salida del report automático de Pivotal Tracker.

```
feature Navigation from Contact to Sign In (Finished)
2014-11-24 10:55PM CET        Alfonso Peralta moved and scheduled
this story before 'Register a User'
2014-11-24 10:55PM CET        Alfonso Peralta moved this story after
'Navigation from Home to Catalog'
2014-11-24 10:58PM CET        Alfonso Peralta moved and scheduled
this story before 'Edit Book from the catalog'
2014-11-24 10:59PM CET        Alfonso Peralta started this feature
2014-11-24 10:59PM CET        Alfonso Peralta finished this feature

feature Navigation from Sign In to Contact (Finished)
2014-11-24 10:55PM CET        Alfonso Peralta moved and scheduled
this story before 'Register a User'
2014-11-24 10:55PM CET        Alfonso Peralta moved this story
after 'Navigation from Contact to Sign In'
2014-11-24 10:58PM CET        Alfonso Peralta moved and scheduled
this story before 'Edit Book from the catalog'
2014-11-24 10:59PM CET        Alfonso Peralta started this feature
2014-11-24 10:59PM CET        Alfonso Peralta finished this feature

feature Navigation from Home to Catalog (Finished)
2014-11-24 10:58PM CET        Alfonso Peralta moved and scheduled
this story before 'Edit Book from the catalog'
2014-11-24 10:59PM CET        Alfonso Peralta started this feature
2014-11-24 10:59PM CET        Alfonso Peralta finished this feature
```

```
feature Navigation from Home to Account (Finished)
2014-11-24 10:58PM CET        Alfonso Peralta moved and scheduled
this story before 'Edit Book from the catalog'
2014-11-24 10:59PM CET        Alfonso Peralta started this feature
2014-11-24 10:59PM CET        Alfonso Peralta finished this feature

feature Navigation from Catalog to Home (Finished)
2014-11-24 10:58PM CET        Alfonso Peralta moved and scheduled
this story before 'Edit Book from the catalog'
2014-11-24 10:59PM CET        Alfonso Peralta started this feature
2014-11-24 10:59PM CET        Alfonso Peralta finished this feature

feature Navigation from Catalog to Account (Finished)
2014-11-24 10:58PM CET        Alfonso Peralta moved and scheduled
this story before 'Edit Book from the catalog'
2014-11-24 10:59PM CET        Alfonso Peralta started this feature
2014-11-24 10:59PM CET        Alfonso Peralta finished this feature

feature Navigation from Account to Home (Finished)
2014-11-24 10:58PM CET        Alfonso Peralta moved and scheduled
this story before 'Edit Book from the catalog'
2014-11-24 11:00PM CET        Alfonso Peralta started this feature
2014-11-24 11:00PM CET        Alfonso Peralta finished this feature

feature Navigation from Account to Catalog (Finished)
2014-11-24 10:58PM CET        Alfonso Peralta moved and scheduled
this story before 'Edit Book from the catalog'
2014-11-24 11:00PM CET        Alfonso Peralta started this feature
2014-11-24 11:00PM CET        Alfonso Peralta finished this feature

feature Logout from Home (Finished)
2014-11-24 11:00PM CET        Alfonso Peralta moved and scheduled
this story before 'Edit Book from the catalog'
2014-11-24 11:00PM CET        Alfonso Peralta started this feature
2014-11-24 11:00PM CET        Alfonso Peralta finished this feature

feature Logout from Catalog (Finished)
2014-11-24 11:00PM CET        Alfonso Peralta moved and scheduled
this story before 'Edit Book from the catalog'
2014-11-24 11:00PM CET        Alfonso Peralta started this feature
2014-11-24 11:00PM CET        Alfonso Peralta finished this feature
```

```
feature Logout from Account (Finished)
2014-11-24 11:00PM CET        Alfonso Peralta moved and scheduled
this story before 'Edit Book from the catalog'
2014-11-24 11:00PM CET        Alfonso Peralta started this feature
2014-11-24 11:00PM CET        Alfonso Peralta finished this feature

feature Add a new Book to the catalog (Finished)
2014-11-24 11:00PM CET        Alfonso Peralta moved and scheduled
this story before 'Edit Book from the catalog'
2014-11-24 11:00PM CET        Alfonso Peralta started this feature
2014-11-24 11:00PM CET        Alfonso Peralta finished this feature
2014-11-24 11:10PM CET        Alfonso Peralta delivered this feature
2014-11-24 11:10PM CET        Alfonso Peralta accepted this feature
2014-11-24 11:10PM CET        Alfonso Peralta finished this feature
2014-11-24 11:10PM CET        Alfonso Peralta delivered this feature
2014-11-24 11:10PM CET        Alfonso Peralta accepted this feature
2014-11-24 11:10PM CET        Alfonso Peralta finished this feature
2014-12-21 08:33PM CET        Alfonso Peralta delivered this feature
2014-12-21 08:33PM CET        Alfonso Peralta accepted this feature
2014-12-21 08:34PM CET        Alfonso Peralta finished this feature

feature Edit Book from the catalog (Finished)
2014-11-24 11:00PM CET        Alfonso Peralta moved and scheduled
this story after 'Add a new Book to the catalog'
2014-11-24 11:00PM CET        Alfonso Peralta started this feature
2014-11-24 11:00PM CET        Alfonso Peralta finished this feature

feature Sign in (Finished)
2014-11-24 10:58PM CET        Alfonso Peralta moved and scheduled
this story after 'Register a User'
2014-11-24 10:59PM CET        Alfonso Peralta started this feature
2014-11-24 10:59PM CET        Alfonso Peralta finished this feature

feature Register a User (Finished)
2014-11-24 10:57PM CET        Alfonso Peralta moved and scheduled
this story
2014-11-24 10:58PM CET        Alfonso Peralta finished this feature

chore Deploy web aplication on Heroku (Accepted)
2014-11-26 01:07PM CET        Alfonso Peralta moved and scheduled
this story after 'Change DB from SQLite3 to PostgreSQL before deploy'
2014-11-26 01:07PM CET        Alfonso Peralta moved this story
after 'Change DB from SQLite3 to PostgreSQL before deploy'
```

```
2014-12-16 06:22PM CET        Alfonso Peralta started this chore
2014-12-16 06:41PM CET        Alfonso Peralta accepted this chore

chore Change DB from SQLite3 to PostgreSQL before deploy (Accepted)
2014-11-26 01:07PM CET        Alfonso Peralta moved and scheduled
this story after 'Create the association "exemplar" '
2014-11-26 01:07PM CET        Alfonso Peralta moved this story
after 'Create the association "exemplar" '
2014-12-16 06:44PM CET        Alfonso Peralta started this chore
2014-12-17 11:45AM CET        Alfonso Peralta accepted this chore

chore Create the association "exemplar" (Accepted)
2014-11-26 01:07PM CET        Alfonso Peralta moved and scheduled
this story
2014-11-26 01:07PM CET        Alfonso Peralta moved this story
after 'Deploy web aplication on Heroku'
2014-11-26 01:07PM CET        Alfonso Peralta moved and scheduled
this story after 'Deploy web aplication on Heroku'
2014-11-27 07:41PM CET        Alfonso Peralta started this chore
2014-11-27 07:41PM CET        Alfonso Peralta accepted this chore

chore Create the views and controller for "exemplars" (Accepted)
2014-11-27 07:41PM CET        Alfonso Peralta moved and scheduled
this story after 'Create the association "exemplar" '
2014-11-27 07:41PM CET        Alfonso Peralta started this chore
2014-12-02 12:04PM CET        Alfonso Peralta accepted this chore
chore Create the views and controller for "exemplars" (Accepted)

2014-11-27 07:42PM CET        Alfonso Peralta moved and scheduled
this story
2014-12-02 12:04PM CET        Alfonso Peralta started this chore
2014-12-02 12:04PM CET        Alfonso Peralta accepted this chore

chore Take a Snapsot of stage 1 (Accepted)
2014-12-02 12:06PM CET        Alfonso Peralta moved and scheduled
this story after 'Create user stories for "My Books" features'
2014-12-16 06:23PM CET        Alfonso Peralta started this chore
2014-12-16 06:41PM CET        Alfonso Peralta accepted this chore

chore Start the project document (Accepted)
2014-12-02 12:07PM CET        Alfonso Peralta moved and scheduled
this story after 'Take a Snapsot of stage 1'
2014-12-16 06:22PM CET        Alfonso Peralta started this chore
```

```
2014-12-16 06:41PM CET        Alfonso Peralta accepted this chore

feature Add an Exemplar to My Books (Finished)
2014-12-03 12:05PM CET        Alfonso Peralta started this feature
2014-12-16 06:41PM CET        Alfonso Peralta finished this feature

feature Remove an Exemplar from My Books (Finished)
2014-12-03 12:06PM CET        Alfonso Peralta started this feature
2014-12-16 06:41PM CET        Alfonso Peralta finished this feature

chore Refactor the features and add new scenarios (Accepted)
2014-12-03 12:55PM CET        Alfonso Peralta started this chore
2014-12-03 12:55PM CET        Alfonso Peralta accepted this chore

feature Create Assistance to Meeting (Finished)
2014-12-16 06:45PM CET        Alfonso Peralta started this feature
2014-12-17 11:45AM CET        Alfonso Peralta finished this feature

feature Cancel Assitance to Meeting (Finished)
2014-12-16 06:45PM CET        Alfonso Peralta started this feature
2014-12-17 11:45AM CET        Alfonso Peralta finished this feature

feature Create New Meeting (Finished)
2014-12-16 06:45PM CET        Alfonso Peralta started this feature
2014-12-17 11:45AM CET        Alfonso Peralta finished this feature

feature View All Meetings (Finished)
2014-12-16 06:45PM CET        Alfonso Peralta started this feature
2014-12-17 11:45AM CET        Alfonso Peralta finished this feature

chore Add the scenario steps as tasks in pivotal tracker (Accepted)
2014-12-17 11:45AM CET        Alfonso Peralta started this chore
2014-12-17 12:01PM CET        Alfonso Peralta accepted this chore

chore Control ISBN validations (both 10 and 13) (Started)
2014-12-21 08:33PM CET        Alfonso Peralta started this chore

chore Hide password from textbox (Started)
2014-12-21 08:59PM CET        Alfonso Peralta started this chore

chore Create user stories for "My Books" features (Accepted)
2014-12-02 12:04PM CET        Alfonso Peralta moved and scheduled
this story
```

```
2014-12-02 12:05PM CET        Alfonso Peralta started this chore
2014-12-02 12:05PM CET        Alfonso Peralta unstarted this chore
2014-12-02 12:06PM CET        Alfonso Peralta started this chore
2014-12-02 12:06PM CET        Alfonso Peralta unstarted this chore
2014-12-16 06:22PM CET        Alfonso Peralta started this chore
2014-12-16 06:41PM CET        Alfonso Peralta accepted this chore
```