

# GOATBooks



Arquitectura e Integración de Sistemas Software

Grado de Ingeniería del Software

Curso 2

Daniel Arellano Martínez (danaremar@alum.us.es)

Eduardo Miguel Botía Domingo (edubotdom@alum.us.es)

Juan Noguerol Tirado (juanogtir@alum.us.es)

Javier Vázquez Zambrano (javvazzam@alum.us.es)

Tutor: Alfonso Márquez

Número de grupo: G1

Enlace de la aplicación: [goatbooks.appspot.com](https://goatbooks.appspot.com)

Enlace de proyecto en projETSII:

<https://projetsii.informatica.us.es/projects/n6q7gqfgwdzqr3xp7rx>

## HISTORIAL DE VERSIONES

Fecha	Versión	Detalles	Participantes
16/03/2019	1.0	Creación para el entregable 1: · Introducción · Creación de proyectos · Despliegue Aplicación Web · Documentación · Mockup · Modelado de la arquitectura del sistema	Daniel Arellano Martínez Eduardo Miguel Botía Domingo Juan Noguerol Tirado Javier Vázquez Zambrano
28/03/2019	2.0	Añadidos para el entregable 2: · Implementación de las páginas de resultados · Consumo de las 4 APIs	Daniel Arellano Martínez Eduardo Miguel Botía Domingo Juan Noguerol Tirado Javier Vázquez Zambrano

# Índice

1	Introducción .....	4
1.1	Aplicaciones integradas .....	6
1.1.1	Wikipedia .....	6
1.1.2	Google Books .....	6
1.1.3	Pixabay.....	6
1.1.4	Here .....	7
1.2	Evolución del proyecto .....	8
2	Prototipos de interfaz de usuario .....	11
2.1	Vista Pantalla Inicial .....	11
2.2	Vista Bibliotecas .....	11
2.3	Vista Libros.....	12
2.4	Vista Búsqueda Principal.....	12
2.5	Vista Resultados Libro.....	13
2.6	Vista Más información .....	13
2.7	Vista Mi Biblioteca .....	14
2.8	Vista Comentarios.....	14
2.9	Vista Contáctanos .....	15
3	Arquitectura .....	16
3.1	Diagrama de componentes.....	16
3.2	Diagrama de despliegue .....	17
3.3	Diagrama de secuencia de alto nivel .....	19
3.4	Diagrama de clases .....	20
	Referencias .....	21

## 1 Introducción

Desde hace unos años, una revolución digital ha llegado a la sociedad cambiando por completo todos los aspectos de nuestra vida, desde el ocio al trabajo y a la ampliación de nuevos horizontes en la industria. Esta, es una tendencia que lejos de representar una burbuja, ha demostrado ser una gran amenaza para aquellos sectores sin presencia en la red, ya sea por inoperatividad o por una estrategia alternativa, quedando rezagados y sin la posibilidad de competir con las mismas bazas frente a su competencia, en una sociedad va camino de ser nativamente digital.

El mercado del entretenimiento ha sufrido un cambio radical gracias a los avances en la tecnología, dando paso a nuevas alternativas de contenido audiovisual que, por lo general, debido a su juventud y necesidad de irrumpir en el mercado, han cambiado las reglas del juego, dejando al sector del libro en una encrucijada, *renovarse o morir*.

Por lo general, este sector ha intentado dar un cambio introduciendo novedades como el libro electrónico, que facilita la distribución del libro, aunque por lo general carece de aceptación a nivel popular, ya que el público objetivo de este sector prefiere el formato físico.

Estudiando esta necesidad por parte del mercado, hemos planteado una aplicación web basada en reunir toda la información sobre un libro introducido por un usuario, para acceder a dicha información desde un único sitio, sin tener que navegar en multitud de sitios web en paralelo para obtenerla. Pese a que defendemos internet como una fuente de conocimiento sin precedentes, la mayor parte de la información, sobre todo histórica, está almacenada en papel. Una fuente de conocimiento que, a nuestro punto de vista, capta menos la atención del usuario medio. Creemos que dar información al usuario, es darle poder de opción y abrirle posibilidades para que pueda enriquecerse de lo que le ofrece el formato físico.

Dado al nivel económico de multitud de personas en nuestro entorno más cercano, la decisión de ofrecerles la mayor cantidad de información posible sobre libros de manera gratuita ha sido una de nuestras mayores prioridades, por lo que en nuestra web podrán ver las bibliotecas o librerías más cercanas que existan a su zona. Se visualizará en un mapa interactivo, en el que podrá localizarse el usuario y la localización de estas.

A un título, o cualquier campo de información que sirva para realizar una búsqueda, se devolverá un listado de los libros que más se acercan a la descripción. A cada resultado, se indexará un enlace que redirija a una página con la información más relevante de dicho libro como su portada, su número de páginas, una reseña, una información del autor principal de la obra, una imagen de este, así como aquella información que pueda resultar relevante al usuario.

Confiamos en la temática de este “*mashup*”, porque vemos el sector escasamente considerado especialmente entre la juventud, que espera a las versiones audiovisuales

de libros para disfrutar de una historia. Creemos que disponer de toda la información accesible y sencilla puede ayudar a facilitar el acceso a la lectura de aquellos no aficionados o introducidos a la lectura. También puede ser una herramienta útil para aquellos usuarios en busca de un libro desconocido por el público general, para reunir información sobre él.

## 1.1 Aplicaciones integradas

Para llevar a cabo nuestra misión, nos apoyamos de cuatro sistemas externas, o aplicaciones integradas que nos aportan la información que acabamos mostrando al usuario, convirtiendo a nuestra plataforma en un intermediario entre ambas partes. Pese a los enésimos cambios que han requeridos imprevistos de muy amplia índole, trabajamos con las aplicaciones que mostramos a continuación:

### 1.1.1 Wikipedia

Wikipedia es una enciclopedia gratuita en línea que ofrece información proveída de sus millones de usuarios en todo el mundo de forma altruista en una gran variedad de idiomas, y que ofrece una enorme cantidad de recursos al visitante.



Su API es usada en nuestro proyecto para obtener información sobre el autor principal de la obra. Por limitaciones de la plataforma, hemos decidido incluir su versión inglesa en nuestro proyecto.

### 1.1.2 Google Books

Google Books es un servicio de la multinacional Google, que ofrece a sus usuarios una base de datos gigantesca, con todo tipo de precios y detalles sobre obras de todo tipo. Pese a ser un único servicio, incluye una enorme cantidad de servicios relacionados con libros, como un buscador, un acceso directo a su punto de venta de libros, o una aplicación de gestión de libros en formato de librerías, con el fin de invitar al usuario a almacenar sus títulos favoritos, sea en papel o en formato digital, clasificados a su gusto.



Su API ofrece una enorme cantidad de recursos y posibilidades, aunque en este caso hemos planteado utilizar utilidades como la búsqueda de libros y la posibilidad de almacenar bibliotecas virtuales, o librerías. Además, facilita un sistema que ayuda a obtener todo tipo de información de su base de datos de libros.

### 1.1.3 Pixabay

Pixabay es un servicio de compartición de imágenes, con ciertas condiciones, compuesta por una comunidad de usuarios razonablemente grande. Ofrecen su material de manera gratuita a todo usuario que acceda a su página web.



Su API nos ofrece un conjunto de imágenes relacionadas con una búsqueda que realice el usuario, según varios parámetros. Usaremos este servicio para ofrecer fotos de los autores, y complementar la información que aporta Wikipedia.

#### 1.1.4 Here

Here es una empresa que ofrece servicios de mapas a empresas, desarrolladores y usuarios a través del servicio *WeGo*. Presenta la ventaja de ser un servicio más asequible que su competencia *Google Maps*. Cuenta con mapas de todo el mundo y con una rica base de datos de todo tipo de negocios y establecimientos, que pueden ser consultados por el usuario.



Su API nos permite mostrar las bibliotecas y establecimientos que tengan relación con los libros cercanos a la ubicación del usuario, o de una localización en coordenadas que el mismo especifique. Junto al listado, el usuario puede tener acceso a un mapa interactivo, en el que el usuario tendrá la posibilidad de consultar la posición de los distintos establecimientos.

## 1.2 Evolución del proyecto

Nombre aplicación	URL documentación API
Wikipedia	<a href="https://en.wikipedia.org/w/api.php">https://en.wikipedia.org/w/api.php</a>
Google Books	<a href="https://developers.google.com/books/?hl=es-419">https://developers.google.com/books/?hl=es-419</a>
Pixabay	<a href="https://pixabay.com/api/docs/">https://pixabay.com/api/docs/</a>
Here	<a href="https://developer.here.com/">https://developer.here.com/</a>

TABLA 1. APLICACIÓN INTEGRADAS

- Evolución del proyecto hasta la entrega del primer entregable.

En un principio, intentamos utilizar las APIs de BookMonch o FeedBooks, pero ambos utilizaban una compresión de tipo .gzip, lo cual causó que finalmente tuviésemos que prescindir de estas, haciéndonos cambiar ligeramente el planteamiento. Tuvimos que utilizar otras APIs para realizar otras, o sustituir las funciones que habíamos plantateado para nuestra aplicación.

En un principio, para localizar las bibliotecas, planteamos el uso de la API de Google Places, sin embargo, planteamos la posibilidad de usar la API que a priori parece que ofrece la Junta De Andalucía que recopila información sobre las bibliotecas en su territorio. Sin embargo, desde el grupo tenemos incertidumbre de que funcione, ya que la documentación no es precisamente abundante. En caso de que fracase, nos veremos obligados a usar la inicialmente planteada.

- Evolución del proyecto hasta la entrega del segundo entregable.

La segunda fase del proyecto, en el que hemos tratado realizar parte de la implementación de los planes que teníamos a posteriori, ha sufrido un gran vuelco cuando hemos presenciado como prácticamente todas las API y servicios que habíamos seleccionado para trabajar, incluidas algunas que reservamos como “suplentes” por si fallaban las titulares que se presentaron a principio del proyecto. Las más destacadas son las que se muestran a continuación.

**GoodReads**, una red social de libros: íbamos a usarla como fuente de reseñas y comentarios, que íbamos a utilizar para postear opiniones, sin embargo, resultó ser inviable porque usaba XML. No obstante, avistamos una oportunidad cuando vimos que el recurso que podía utilizarse para obtener la valoración de una obra por los usuarios, recurso rating, ya que este era el único recurso en JSON que la empresa mantenía activo. Realizamos pruebas y contemplamos como a simple vista, parecía



que la aplicación funcionaba. Las siguieron la implementación de los *resource*, y los controladores. Cuando fuimos a probar la interfaz visual vimos como no funcionaba, y caímos en la cuenta de que el servicio no devuelve respuestas en formato JSON correctamente. Intentamos *parsear* el resultado, pero se hizo imposible en nuestra inexperiencia.

**OpenLibrary**, la base de datos de libros: Tras estudiar su documentación y realizar algunas pruebas que a simple vista parecían satisfactorias, parecía que se convertiría en la fuente de información del proyecto, hasta que nos dimos cuenta que la antigua API JSON fue descontinuada, y aunque sigue funcionando una versión moderna, cuya base es XML, aunque admite en unas pocas consultas el formato JSON, esta devuelve archivos JSON mal escritos, ya que identifica los ISBN como nombre de la propiedad ISBN, por lo cual no se pueden *parsear* las respuestas que ofrece el servicio a las búsquedas o consultas. Pese implementar todas las capas necesarias para implementarla, los intentos de *parsearlo* resultaron ser infructuosos.

**BookMooch** : Pensada para obtener información, comentarios y realizar búsquedas de libros tras errar las diferentes alternativas que planteamos, al tratar de serializar a POJO las respuestas no es posible tratarlas. En las búsquedas devuelve, teóricamente, elementos del tipo JSONArray. Recalcamos el teóricamente, ya que la aplicación ofrece una documentación muy pobre, y únicamente una breve introducción a cada método y un ejemplo. Tras intentar implementarlo usando la metodología con la que se implementó un caso muy similar en clase de prácticas, y horas de intentar depurar el código y buscar información, no conseguimos hacerlo funcionar.

**Open Libra**, una librería sin copyright : Contemplando los errores que obteníamos con los anteriores servicios, planteamos usarla como base de datos de libros sin copyright, así como para ampliar el rango de búsqueda del resto de plataformas. Pensamos que la aplicación ofreciese una solución que le permitiese elegir realizar búsquedas de obras publicadas libremente o bajo derechos de autor. Tras tratar de crear sus recursos, descubrimos q únicamente devuelve JSON-P, es decir, ejecutables por JavaScript, y no JSON estándar que consume nuestro servicio.

**Google Places y Maps**, la solución universal para mapas e información de establecimientos: Proporciona una API que constituye una solución casi universal para cualquier usuario en su ámbito. Sin embargo, la dependencia con Google con su API de libros, junto al límite de 1 consulta gratuita al día, invitaron a buscar una alternativa.

**ISBNdb**, la base de datos de libros: Es una solución lógica que un usuario se plantea cuando trata de realizar cualquier aplicación que tenga algo que ver con los libros. Como revela su nombre, contiene una base de datos global con una enorme cantidad de información y recursos sobre libros. Hasta hace relativamente poco, ofrecían un servicio gratuito de API, pero como empresa decidieron restringir el acceso a la misma cobrando.

**Twitter** : Planteamos el uso de su API, pero caímos en la cuenta de que es OAUTH-1, cuya implementación se antojaba un reto al no haber estudiado el estándar en la asignatura. Finalmente, para no retrasar plazos, decidimos usar el *toggle* para páginas web que ofrece la web social para mostrar las notificaciones y novedades del servicio en el perfil del mismo, soterrando las posibilidades de utilizar la API.

**Junta de Andalucía**, información de bibliotecas: El gobierno autonómico puso complicado desde un principio el acceso a sus recursos, pero una vez encontrados, confiamos en el acceso a los mismos tras hacer algunas pruebas. Era el servicio que planteamos desde un inicio para que nos proporcionase información sobre los emplazamientos donde se localiza las bibliotecas y librerías públicas de todos los municipios de la comunidad autónoma. Tras intentar implementar sus *resorces*, caímos en la cuenta de que, pese a ser formato JSON, lo ofrece de manera extraña, con URL como parámetros que varían según el dato que se presenta, o la biblioteca en concreto, las cuales no existían. Tras no conseguir *parsear* los datos, pasamos a buscar otra alternativa.

Por si no tuviésemos pocos problemas con las diferentes aplicaciones sobre las que íbamos a sostener el proyecto, más de 9 API, tras encontrar diferentes alternativas, al desplegar el proyecto en Google Cloud, descubrimos que ninguna funciona en la web, excepto el mero mapa de Here.

La particularidad, es que exactamente el mismo código funciona en local, y al buscar información en el Log que ofrece Google Cloud, el error radica en que las aplicaciones presentan problemas para comunicarse con el servidor.

Además de estos errores, como es común en cualquier proyecto, hemos cometido fallos y errores de todo tipo, con el consecuente tiempo para tratar de depurar el código.

## 2 Prototipos de interfaz de usuario

### 2.1 Vista menú principal.

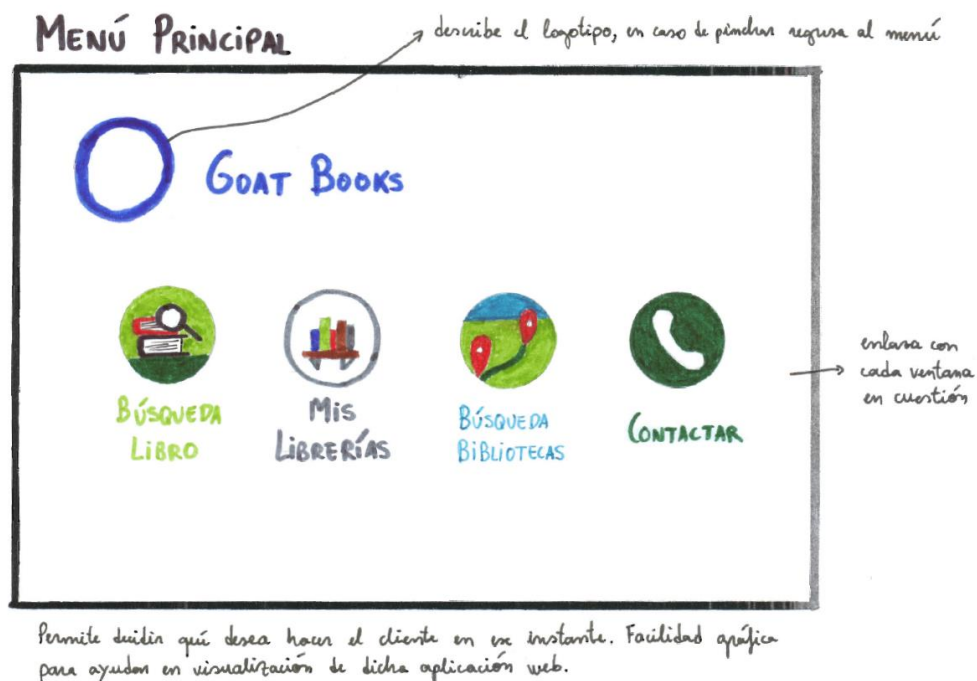


FIGURA 1

### 2.2 Vista búsqueda libro.

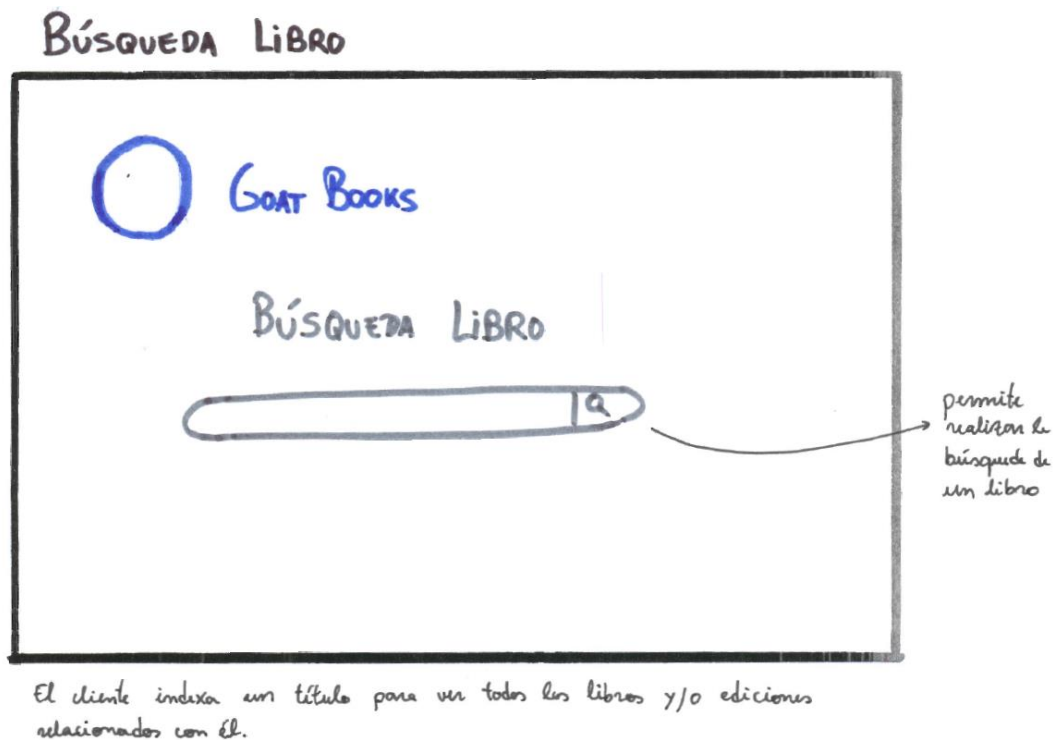


FIGURA 2

## 2.3 Vista resultado de búsqueda de libros.

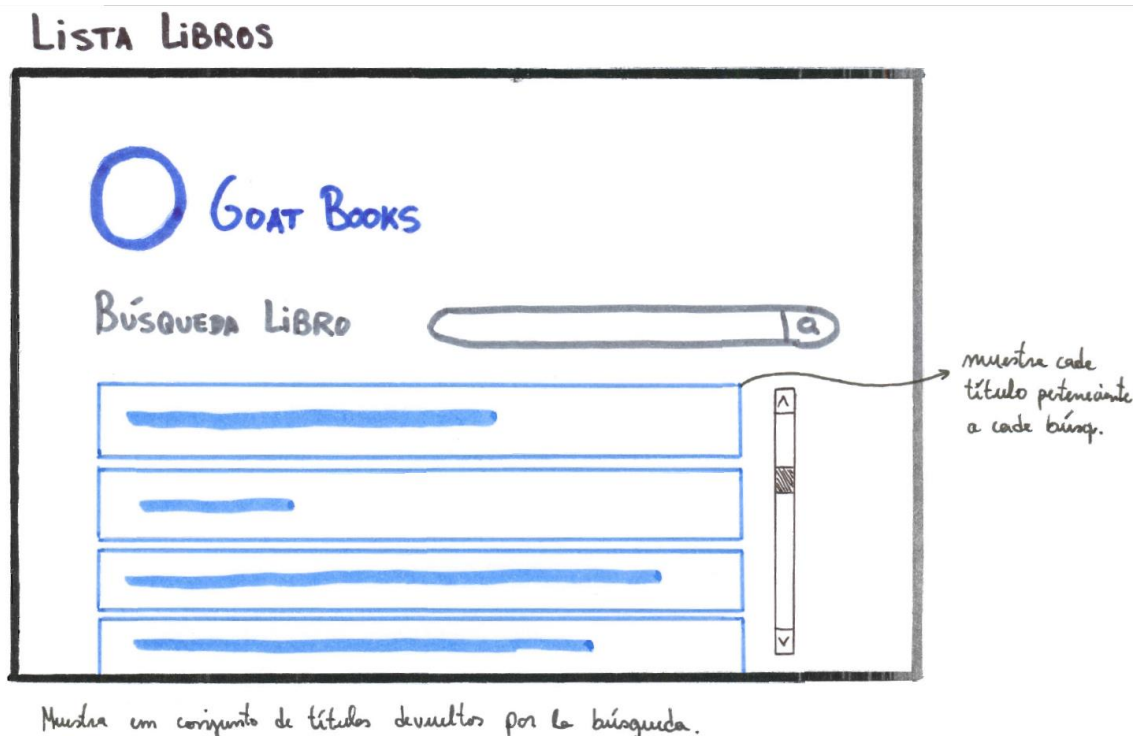


FIGURA 3

## 2.4 Vista detalle libros.

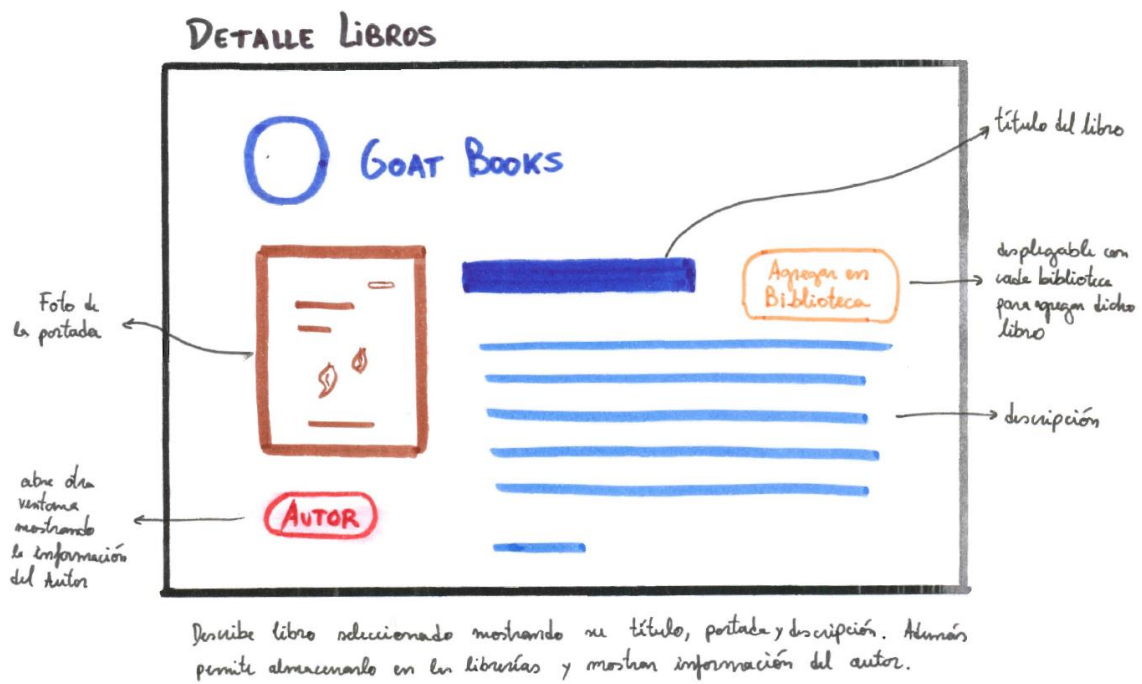


FIGURA 4

## 2.5 Vista información autor.

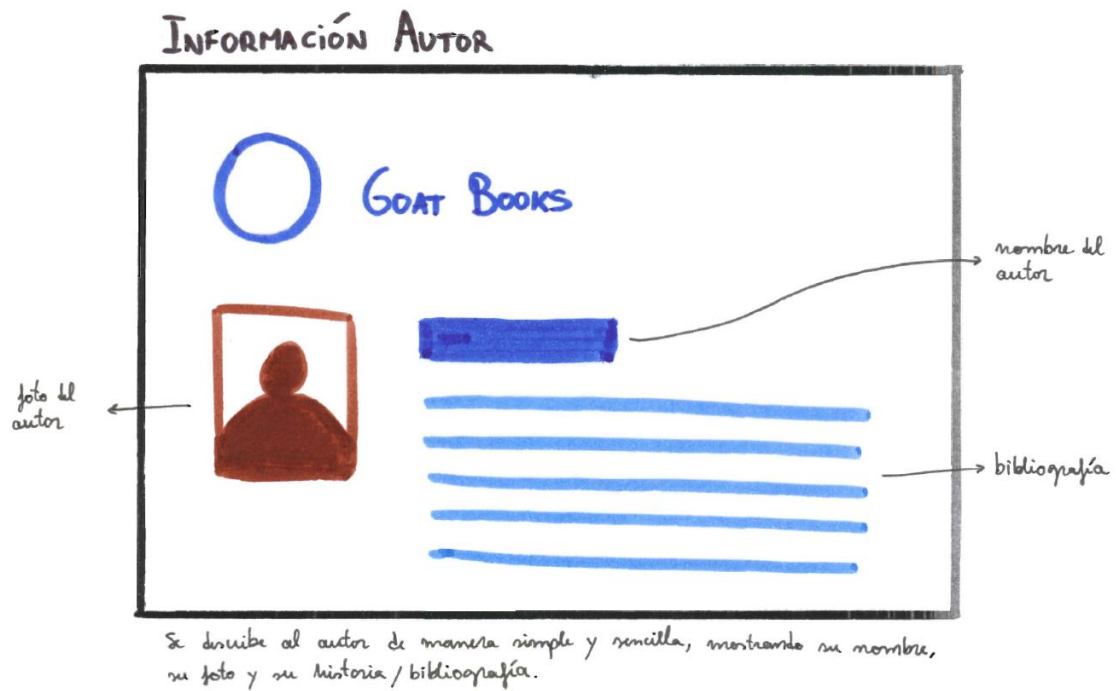


FIGURA 5

## 2.6 Vista librerías del usuario.

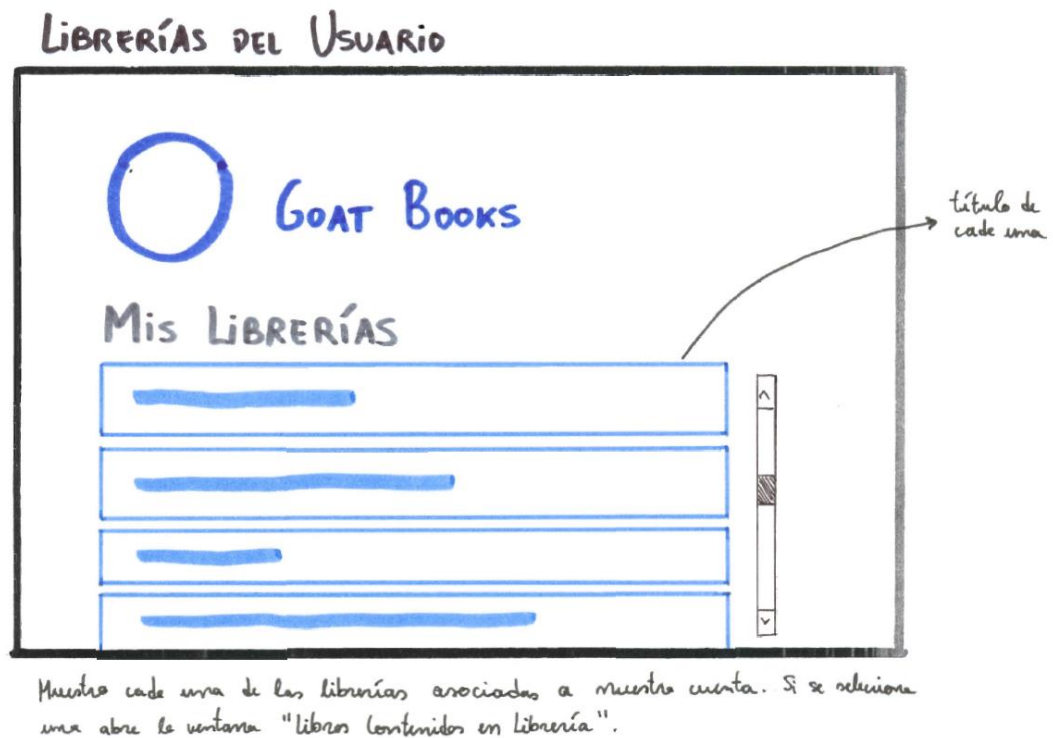
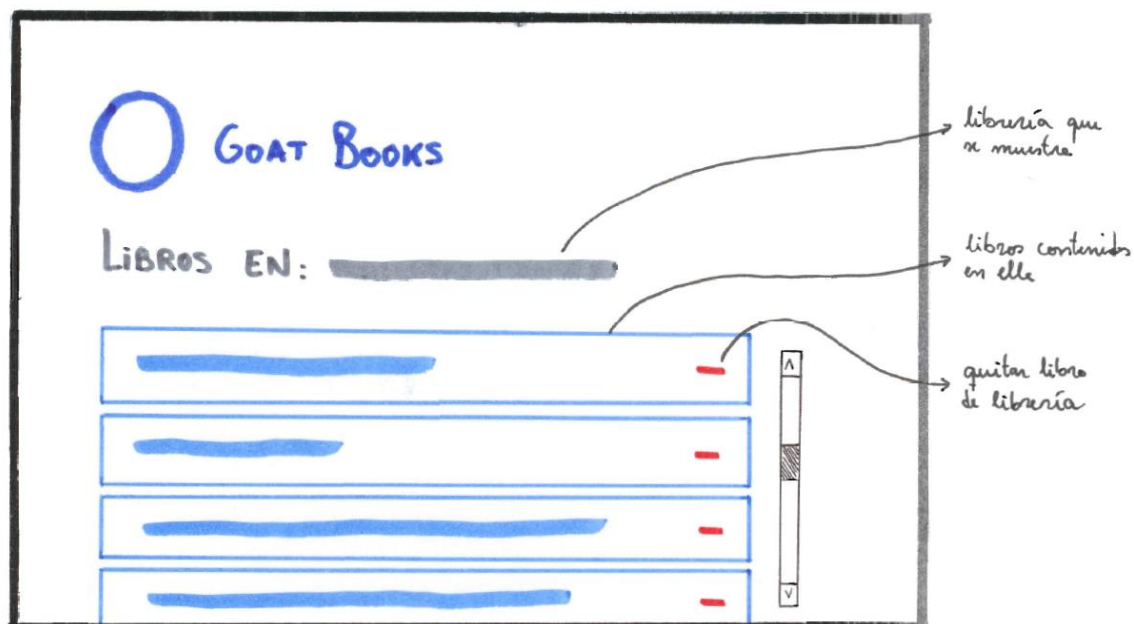


FIGURA 6

## 2.7 Vista libros contenidos en librería.

### LIBROS CONTENIDOS EN LIBRERÍA

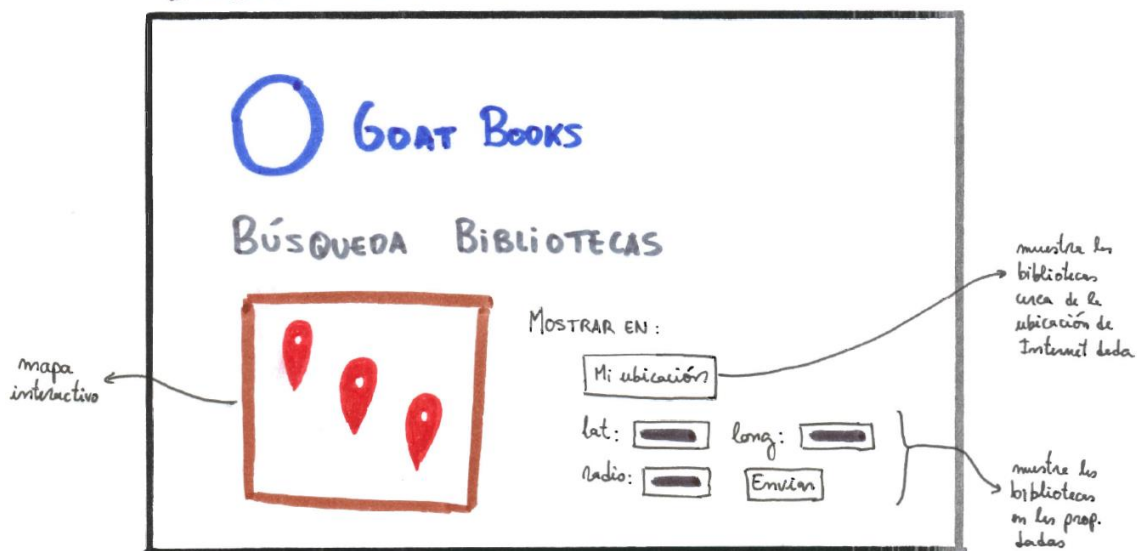


Describe cada una de las librerías mostrando los libros que incluye en su interior y permitiendo eliminarlos de dicha librería en cuestión.

FIGURA 7

## 2.8 Vista Comentarios

### BÚSQUEDA BIBLIOTECAS



Muestra un mapa en el que se posicionan las bibliotecas dadas, según los parámetros dados por el usuario.

FIGURA 8

## 2.9 Vista Contáctanos

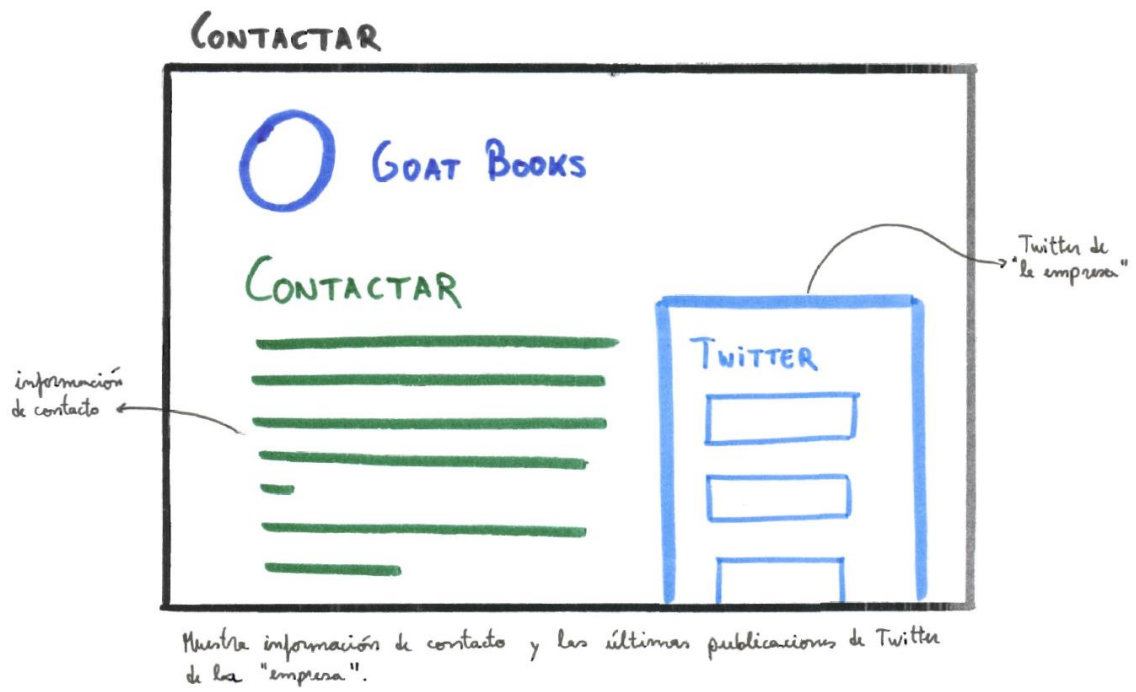
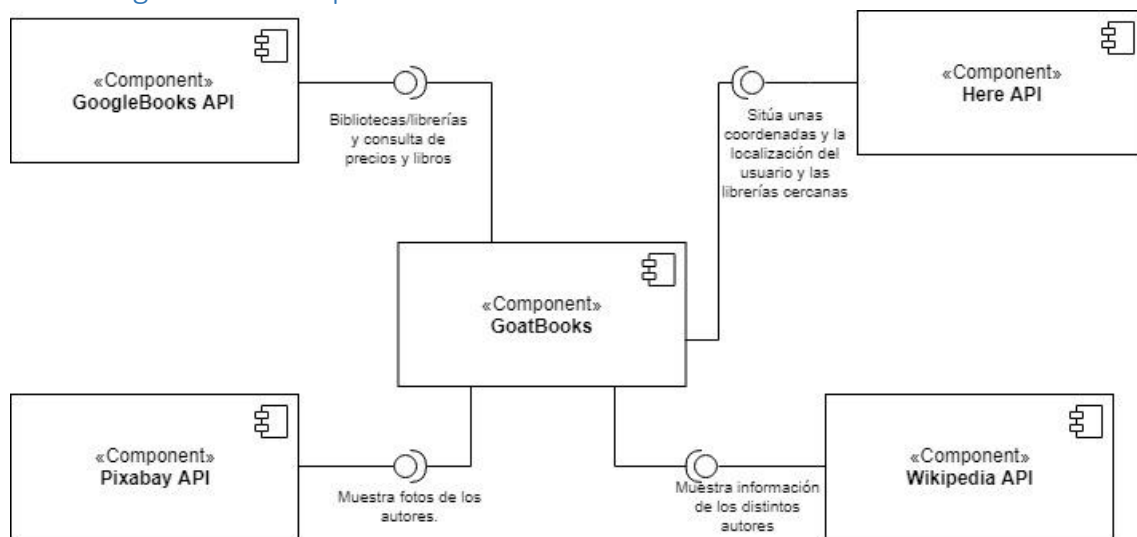


FIGURA 9

## 3 Arquitectura

### 3.1 Diagrama de componentes



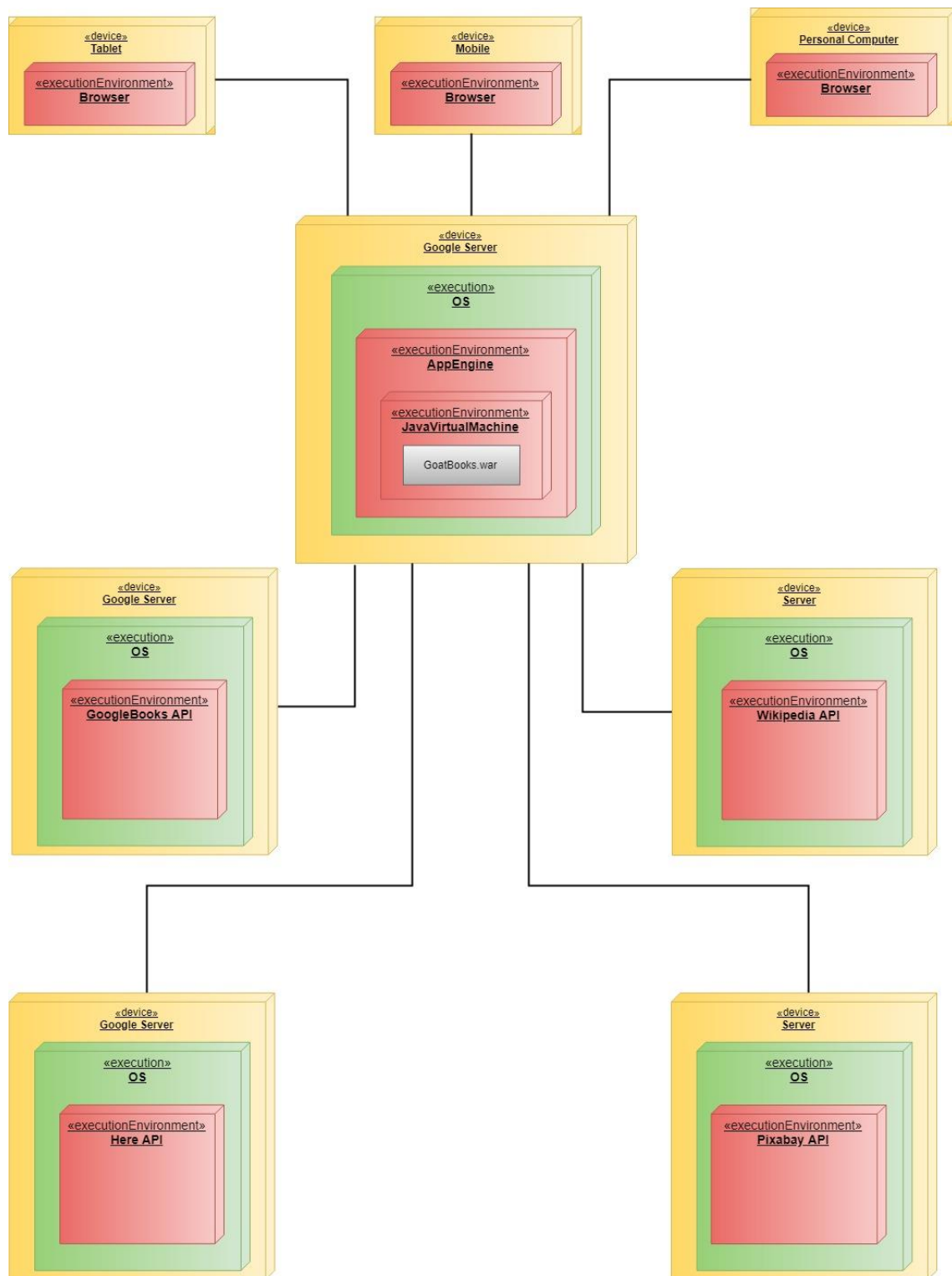
En este diagrama de componentes se detallan la funcionalidad de las APIs a tratar. Usaremos 5 APIs, cuya funcionalidad para el proyecto sería:

- **Wikipedia API**: se realizarán las consultas de la información de los autores, que nos mostrará una pequeña previsualización de la misma.
- **GoogleBooks API**: se establecerá una biblioteca con los libros "Favoritos", además también suministrará información de libros y de precios en caso de que ISBNdb no encuentre el libro deseado.
- **Pixabay API**: nos ofrece un repositorio de fotos para una búsqueda, las cuales usaremos para mostrar fotografías de los autores.
- **Here API**: ubica las distintas Bibliotecas de una ubicación que podrá ser indicada a mano o según la posición actual del usuario.

Todas ellas unidas se unen a una interfaz, se interconectan entre sí y forman la aplicación web de GoatBooks.



### 3.2 Diagrama de despliegue



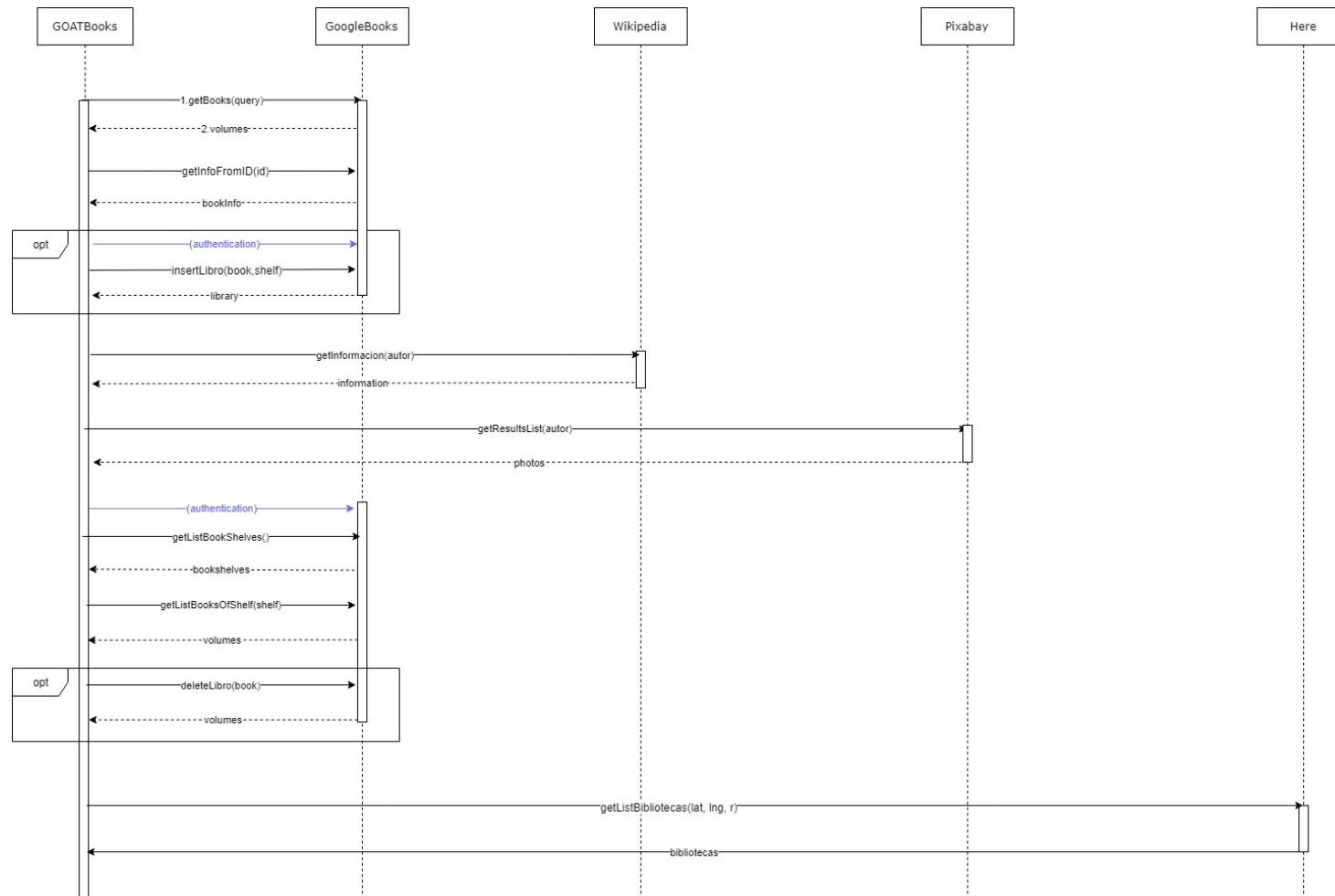
Este diagrama es utilizado para representar la arquitectura física y software sobre la que un sistema informático es desplegado.

Hemos incluido información sobre el despliegue de las API que vamos a utilizar, en concreto su servidor, sistema operativo, y la API del servicio que consultamos o que utiliza la aplicación.

En la parte central y como pilar principal, incluimos el soporte sobre el que se planea que la aplicación funcione, en nuestro caso, la plataforma AppEngine, junto a la máquina virtual de Java, que correrá la aplicación.

Finalmente, incluimos en el diagrama, en su parte superior, a los distintos perfiles de usuarios del sitio, como pueden ser formato móvil, de escritorio y tableta.

### 3.3 Diagrama de secuencia de alto nivel



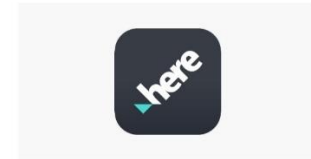
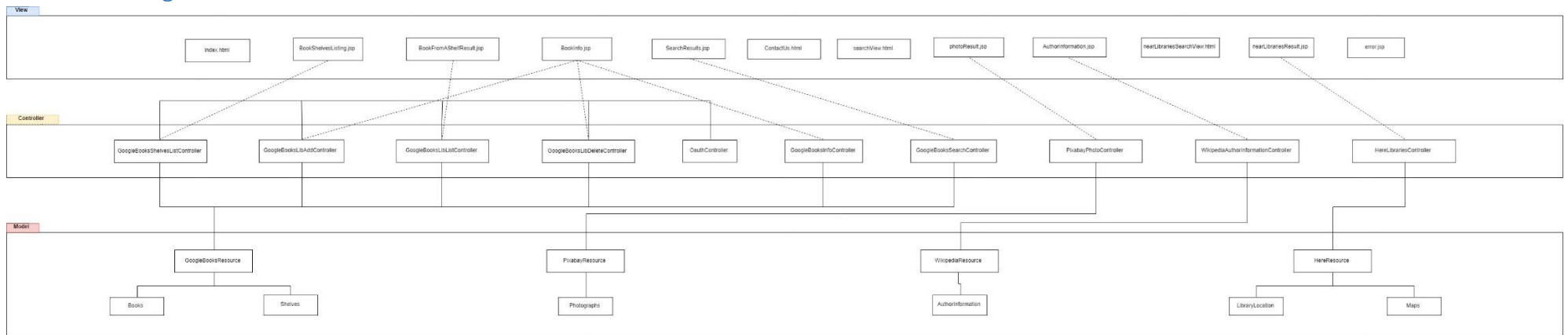
La aplicación usa 4 servicios, interaccionando de la manera que se muestra en el diagrama.

Google Books es usado para hacer una búsqueda dada una query facilitada por el usuario. Una vez devueltos los libros, la aplicación hace una petición al servicio para obtener la información sobre el libro, que será mostrada al usuario. Otra opción es gestionar los libros y librerías de la cuenta del usuario, con las peticiones que se muestran.

Una vez mostrada la información, se le da la posibilidad al usuario de que la aplicación le de información sobre el autor, que será requerida a Wikipedia, para la información, y a PixaBay, para una serie de imágenes del mismo.

La integración con Here consiste en hacer consultas al servicio sobre las librerías alrededor del usuario, o dada unas coordenadas concretas. Una vez recibidas, se añaden al mapa por pantalla que le aparece al usuario.

### 3.4 Diagrama de clases



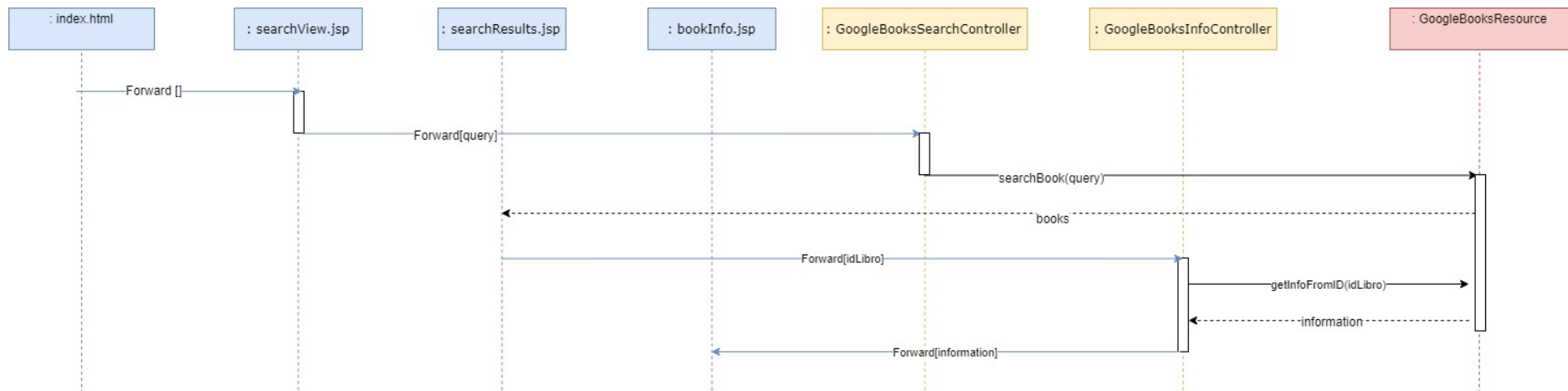
En este diagrama mostramos la interacción entre las diferentes clases dentro del modelo, vista, controlador de cada una de los 4 servicios que implementa GOATBooks.

Destaca Google Books, que posee un mayor número de controladores y es la que presenta una mayor presencia en las vistas al usuario, mientras que PixaBay y Wikipedia presentan una menor presencia, con un solo controlador y vista cada uno de ellos.

Here es un caso similar a los anteriores, aunque presenta la peculiaridad de que parte de su implementación consiste en un JavaScript para implementar el mapa.

### 3.5 Diagramas de secuencia

#### 3.5.1 Diagrama de secuencia. Información Libro.



Para la tarea de mostrar información al usuario, este diagrama muestra la serie de interacciones entre los distintos niveles del modelo-vista-controlador.

En primer lugar, se realiza una búsqueda de los libros que coinciden con el patrón de texto que introduce el usuario. Una vez listados, al pulsar sobre uno de ellos se redirige a una pantalla con la información detallada de las obras y sus autores.

### 3.5.2 Diagrama de secuencia. Información extendida sobre el autor.

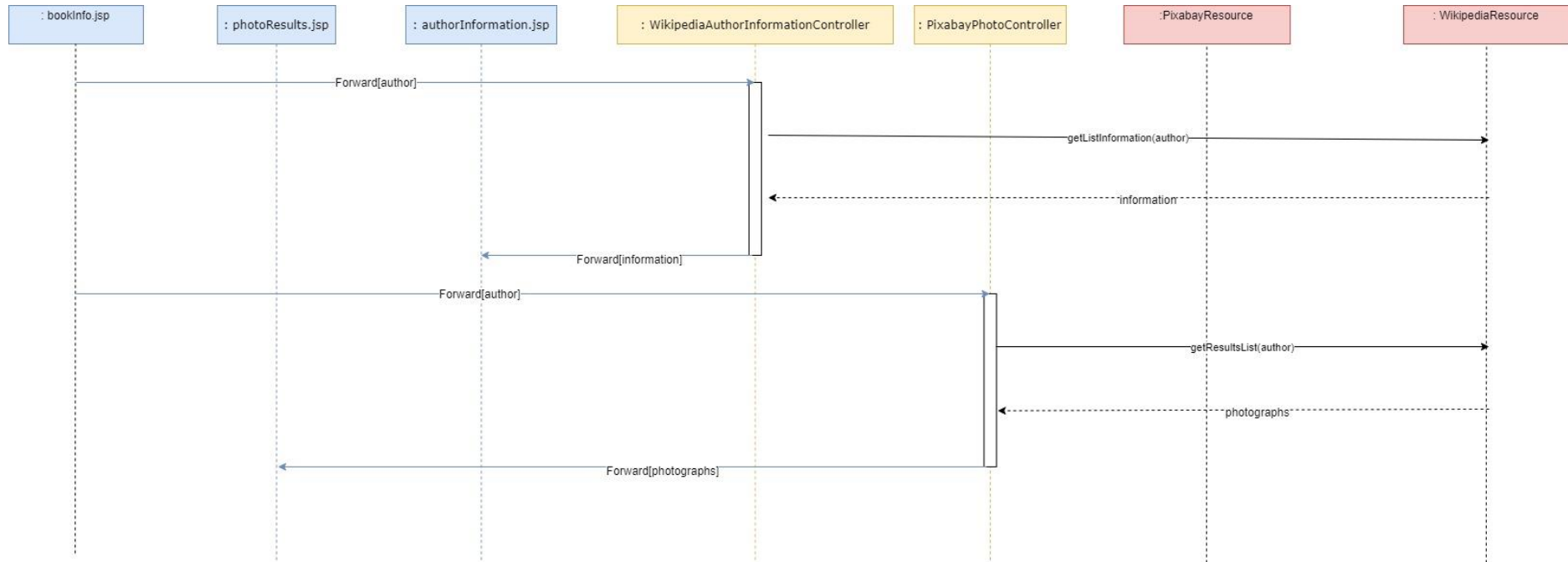


Diagrama en el que se muestra el funcionamiento de la aplicación cuando el usuario requiere más información sobre el autor.

Para mostrar información extendida del autor de la obra, se requiere la interacción del usuario pulsando el botón correspondiente a la información que quiera consultar, bien sea las imágenes sobre el autor disponibles a una pantalla con información del mismo, localizado en la pantalla de información detallada del libro.

### 3.5.3 Diagrama de secuencia. Localizar librerías.

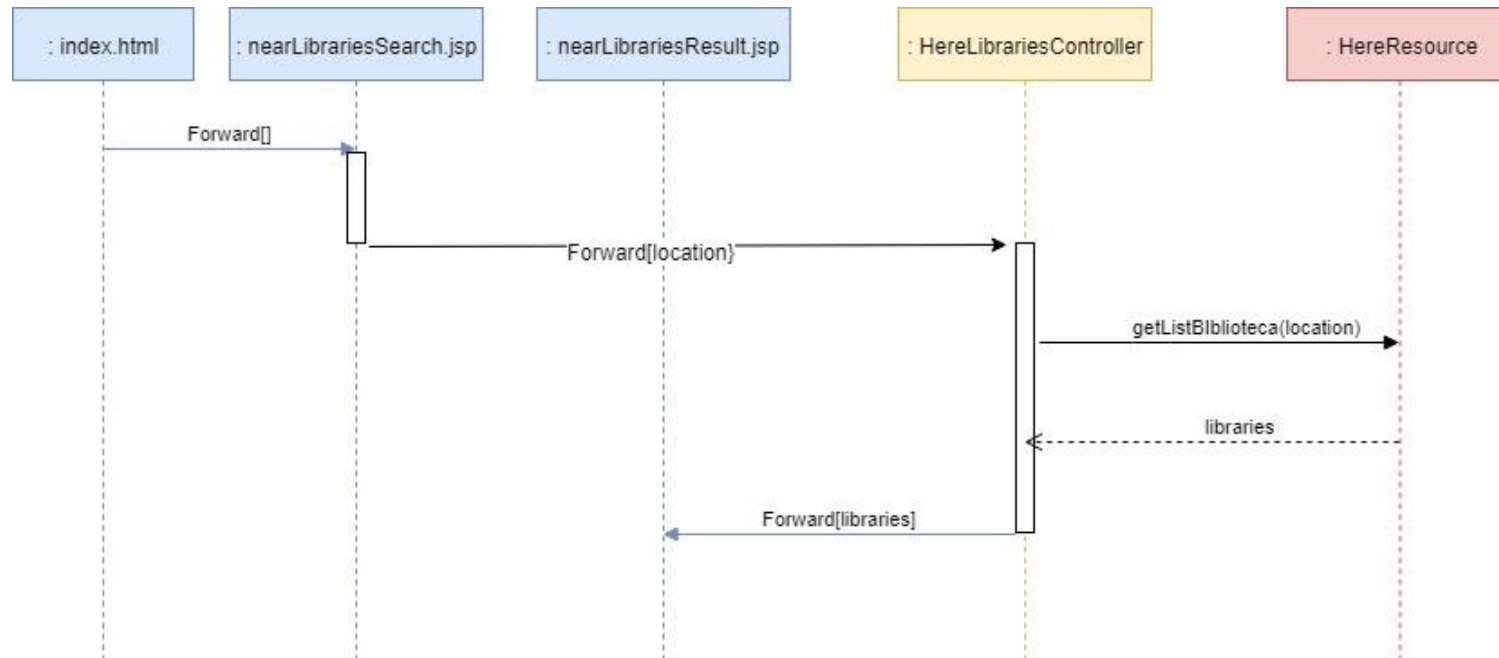
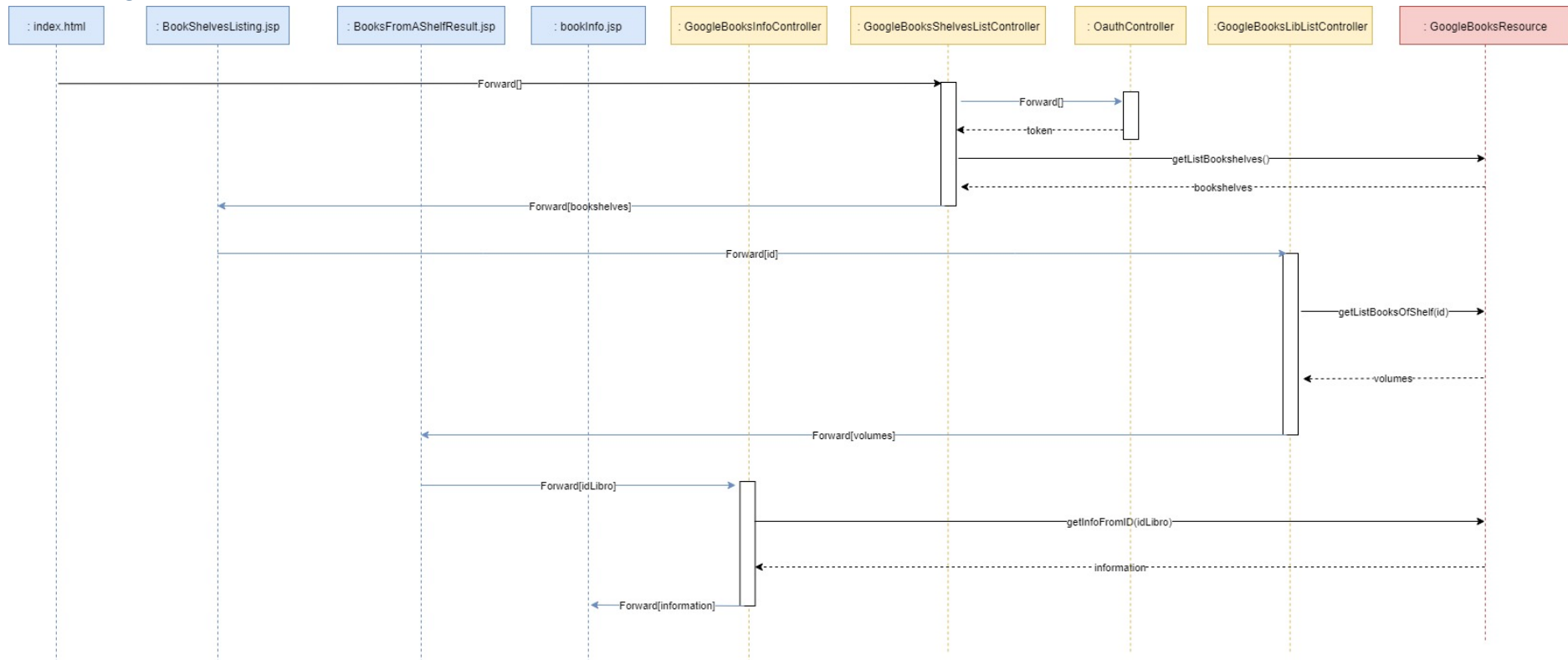


Diagrama en el que se muestra el proceso durante el cual la aplicación obtiene la información sobre la posición de las librerías alrededor del punto indicado por el usuario.

Para acceder a dicha información, se le consulta a Here realizando una consulta indicando los parámetros necesarios para la misma, la longitud, latitud y radio de búsqueda. Los datos retornados serán utilizados para añadir puntos al mapa donde se localicen las librerías.

### 3.5.4 Diagrama de secuencia. Gestión de librería.

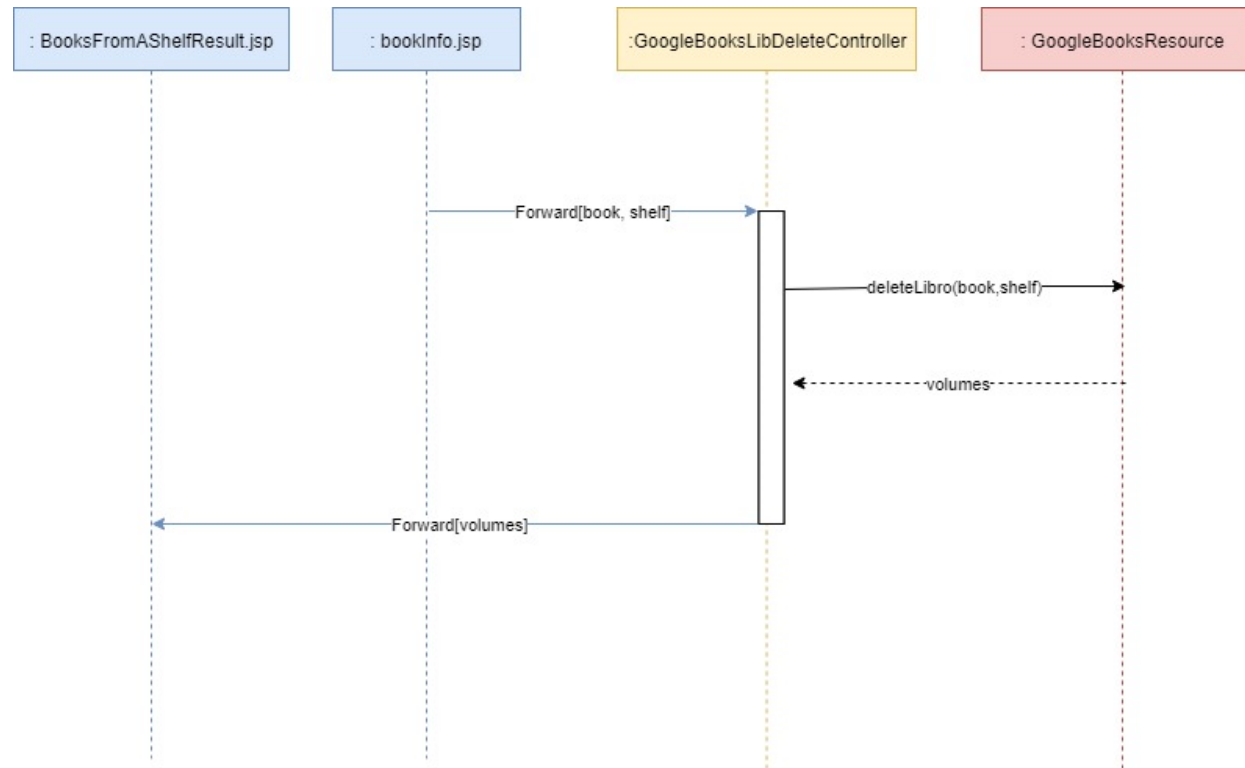


En este diagrama esquematizamos el funcionamiento de la gestión de librerías y libros de la cuenta del usuario.

En primer lugar, previa autenticación obligatoria y concedidos los permisos a la aplicación, esta muestra al usuario una vista con sus librerías, pudiendo elegir una para visualizar los libros que contienen y con la posibilidad de consultar más información acerca de cada uno de los libros y sus autores.

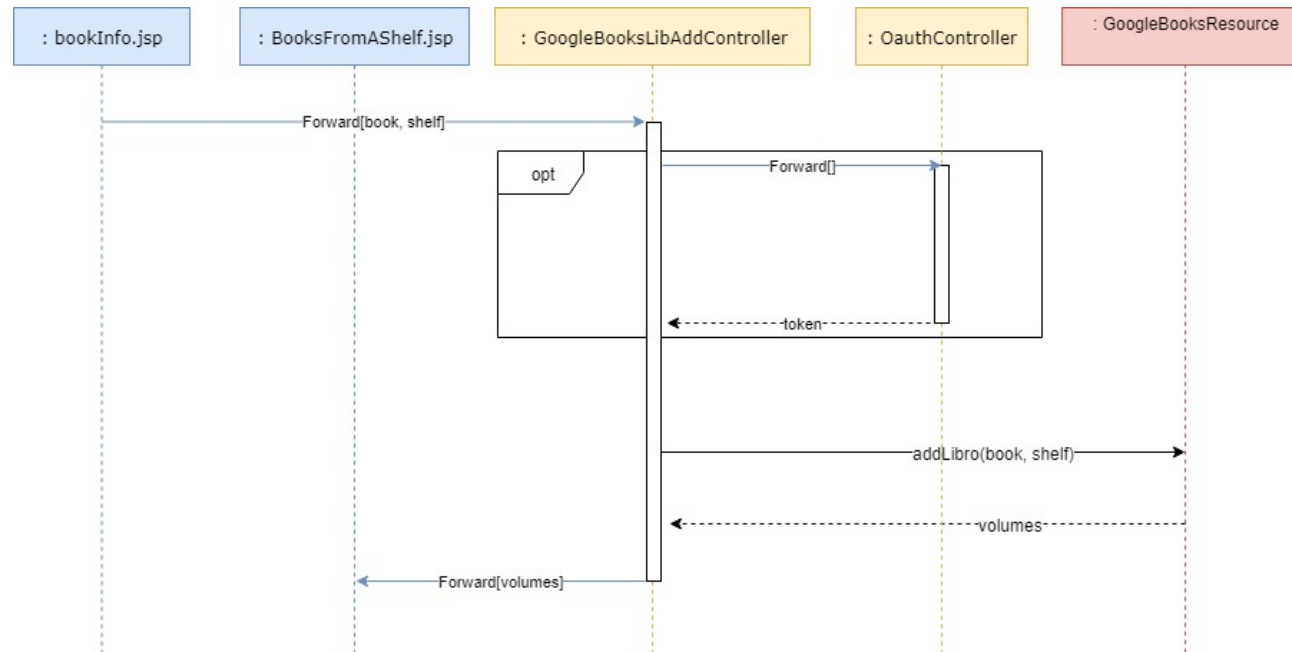


### 3.5.5 Diagrama de secuencia. Eliminación de libro.



Este diagrama muestra el funcionamiento de la eliminación de una obra de una librería del usuario. Para acceder a la vista `BooksFromAShelfResult` debe haberse autenticado anteriormente, por lo que no es necesario implementarlo en este nivel. A nivel de implementación, consiste en un POST que se gestiona desde el resource de Google Books, que recibe la petición del controlador asociado, el cual recibe una petición de la vista del usuario, en la cual el usuario desencadena la acción. Una vez realizada la acción, se redirige al usuario a la página con los libros de la librería que ha añadido, para que compruebe que su operación ha sido realizada con éxito.

### 3.5.6 Diagrama de secuencia. Adición de libro.



Este diagrama muestra el funcionamiento de la aplicación cuando el usuario trata de añadir una obra a su librería, acción para la cual es necesario realizar una autenticación con Google Books, que puede haberse realizado en alguna otra vista, razón para considerarse como operación condicional. Este método, de manera análoga a eliminación, implementa una operación POST en el resource de Google Books. Una vez realizada la acción, se redirige al usuario a la página con los libros de la librería que ha añadido, para que compruebe que su operación ha sido realizada con éxito.

## 4 Implementación

La implementación de este proyecto ha estado marcada por el elevado número de servicios que teníamos planteado implementar y que por una u otra razón acabaron fallando e hicieron perder tiempo y esfuerzo intentando parsear resultados en XML, JSONArray escritos en formato incorrecto, entre otros errores detallados en otro apartado de esta documentación; obligando a una remodelación de la lógica del proyecto que no esperamos desde un principio.

Consideramos que hemos logrado implementar 4 servicios de una manera sencilla y transparente aportando comodidad e intuitividad al usuario. Como detallamos desde el principio, se busca agilizar la experiencia de búsqueda en varios servicios en paralelo para recabar información sobre una obra y su autor, y esta implementación busca este propósito, mediante una interfaz agradable que atraiga al usuario potencial de la aplicación.

Como parte de este intento de ofrecer una aplicación sencilla y no engorrosa en su uso, los únicos momentos en los que el usuario debe usar su teclado son para realizar una búsqueda de una obra, para elegir la librería en la que almacenar una obra y para introducir unas coordenadas concretas de búsqueda para localizar librerías, teniendo en cuenta que podría buscarlas según su ubicación, y ahorrar este paso.

Como parte de la implementación, debimos estudiar el funcionamiento de la API de Google Books que requiere OAuth2, lo que presentó una serie de inconvenientes iniciales y que frenó el desarrollo del proyecto. Finalmente, fue posible la implementación y conseguimos que funcionase como esperábamos.

Debido a las limitaciones de la API de Wikipedia, la búsqueda de información solo arroja una pequeña línea de información y en inglés. Este último inconveniente surge de que las búsquedas a la API en español no devolvían información concluyente, además que el servicio se ofrece en los idiomas francés, alemán e inglés. Intentamos utilizar un conversor para traducir los resultados en XML (que sí contienen información más detallada y en el idioma adecuado), pero resultó fallar más de lo adecuado y decidimos no continuar su implementación.

Para la localización de librerías, invertimos una elevada cantidad de tiempo estudiando como implementar el código JavaScript de los mapas de Here, además de trabajar con parámetros de JS para implementar la funcionalidad de añadir puntos de forma

automática con los resultados que arroja la consulta a los servidores de Here, además de la propia localización. También existe la posibilidad de localizar al usuario en el mapa, función que requirió un estudio previo del componente de HTML que habilitaba el procesamiento de localización del usuario.

También requirió esfuerzos la implementación del toggle de Twitter, que permite una interacción directa con el perfil del equipo de soporte. Al principio, pensamos en implementar una librería externa recomendada por el equipo de Twitter, aunque acabamos prescindiendo de ella. Como solución, implementamos un toggle en formato JavaScript que funciona correctamente, y que es compatible con una mayor cantidad de dispositivos.

## 5 Pruebas

Para el apartado de pruebas hemos planteado hacer pruebas sobre todos aquellos métodos que no implementen OAUTH, de manera que la mayor parte sean automáticas, quedando como manuales aquellas que no sea posible implementar de manera automatizada.

Resumen	
Número total de pruebas realizadas	11
Número de pruebas automatizadas	7 (63,63%)

- Pruebas relacionadas a Google Books.

ID	<b>Prueba 1 – Búsqueda de libros - Google Books</b>
Descripción	Prueba para la detección de errores al implementar búsquedas en Google Books usando servicios RESTful.
Salida esperada	Los datos devueltos en formato JSON son mapeados a una clase Java y a continuación se muestra si la prueba ha sido exitosa.
Resultado	<b>EXITO</b>
Automatizada	Sí, utilizando JUnit.

ID	<b>Prueba 2 – Información detallada de libro - Google Books</b>
Descripción	Prueba para la detección de errores al implementar un acceso a la información detallada de una obra usando un ID de la plataforma Google Books, usando servicios RESTful.
Salida esperada	Los datos devueltos en formato JSON son mapeados a una clase Java y a continuación se muestra si la prueba ha sido exitosa.
Resultado	<b>EXITO</b>
Automatizada	Sí, utilizando JUnit.

<b>ID</b>	<b>Prueba 9 – Consulta de librerías de usuario- Google Books</b>
<b>Descripción</b>	Prueba para la detección de errores al implementar un acceso a las librerías almacenadas por un usuario en su perfil en la plataforma Google Books, usando servicios RESTful.
<b>Salida esperada</b>	Los datos devueltos en formato JSON son mapeados a una clase Java y procesados y mostrados por pantalla, para que el usuario que realiza la prueba pueda comprobar su funcionamiento.
<b>Resultado</b>	<b>EXITO</b>
<b>Automatizada</b>	No, realizada manualmente.

<b>ID</b>	<b>Prueba 10 –Libros pertenecientes a una librería - Google Books</b>
<b>Descripción</b>	Prueba para la detección de errores al implementar un acceso a los libros almacenados por un usuario en una de las librerías que un usuario posea en su perfil en la plataforma Google Books, usando servicios RESTful.
<b>Salida esperada</b>	Los datos devueltos en formato JSON son mapeados a una clase Java y procesados y mostrados por pantalla, para que el usuario que realiza la prueba pueda comprobar su funcionamiento.
<b>Resultado</b>	<b>EXITO</b>
<b>Automatizada</b>	No, realizada manualmente.

<b>ID</b>	<b>Prueba 11 –Añadir libros a una librería - Google Books</b>
<b>Descripción</b>	Prueba para la detección de errores al implementar una adición de un libro a una las librerías que un usuario posea en su perfil en la plataforma Google Books, usando servicios RESTful.
<b>Salida esperada</b>	Los datos devueltos en formato JSON son mapeados a una clase Java y procesados y mostrados por pantalla, para que el usuario que realiza la prueba pueda comprobar su funcionamiento.
<b>Resultado</b>	<b>EXITO</b>
<b>Automatizada</b>	No, realizada manualmente.

<b>ID</b>	<b>Prueba 11 –Eliminar libros a una librería - Google Books</b>
<b>Descripción</b>	Prueba para la detección de errores al implementar una eliminación de un libro a una las librerías que un usuario posea en su perfil en la plataforma Google Books, usando servicios RESTful.
<b>Salida esperada</b>	Los datos devueltos en formato JSON son mapeados a una clase Java y procesados y mostrados por pantalla, para que el usuario que realiza la prueba pueda comprobar su funcionamiento.
<b>Resultado</b>	<b>EXITO</b>
<b>Automatizada</b>	No, realizada manualmente.

- Pruebas relacionadas a Here

<b>ID</b>	<b>Prueba 3 – Búsqueda de bibliotecas - Here</b>
<b>Descripción</b>	Prueba para la detección de errores al implementar búsquedas de librerías en un radio concreto, usando servicios RESTful.
<b>Salida esperada</b>	Los datos devueltos en formato JSON son mapeados a una clase Java y a continuación se muestra si la prueba ha sido exitosa.
<b>Resultado</b>	<b>EXITO</b>
<b>Automatizada</b>	Sí, utilizando JUnit.

<b>ID</b>	<b>Prueba 4 – Búsqueda de bibliotecas - Here</b>
<b>Descripción</b>	Prueba para la detección de errores al implementar búsquedas de librerías, parseando el resultado a una lista de elementos de tipo librería, en un radio concreto, usando servicios RESTful.
<b>Salida esperada</b>	Los datos devueltos en formato JSON son mapeados a una clase Java y a continuación se muestra si la prueba ha sido exitosa.
<b>Resultado</b>	<b>EXITO</b>
<b>Automatizada</b>	Sí, utilizando JUnit.

<b>ID</b>	<b>Prueba 8 – Inserción de puntos con librerías en el mapa - Here</b>
<b>Descripción</b>	Prueba para la detección de errores al implementar la adición de puntos al mapa donde se encuentran las librerías que arroja la búsqueda, usando servicios RESTful para la obtención de resultados y JavaScript para el tratamiento de los datos y adición al mapa en dicho lenguaje.
<b>Salida esperada</b>	Los datos devueltos en formato JSON son mapeados a una clase Java, a partir de la cual se muestran por pantalla los resultados de la búsqueda según la localización, para después ser procesados dichos resultados para ser añadidos mediante JavaScript al mapa.
<b>Resultado</b>	<b>EXITO</b>
<b>Automatizada</b>	No, prueba realizada de forma manual.



- Pruebas relacionadas a Wikipedia.

<b>ID</b>	<b>Prueba 5 – Búsqueda de información - Wikipedia</b>
<b>Descripción</b>	Prueba para la detección de errores al implementar búsquedas de información sobre autores de obras, usando servicios RESTful.
<b>Salida esperada</b>	Los datos devueltos en formato JSON son mapeados a una clase Java y a continuación se muestra si la prueba ha sido exitosa.
<b>Resultado</b>	<b>EXITO</b>
<b>Automatizada</b>	Sí, utilizando JUnit.

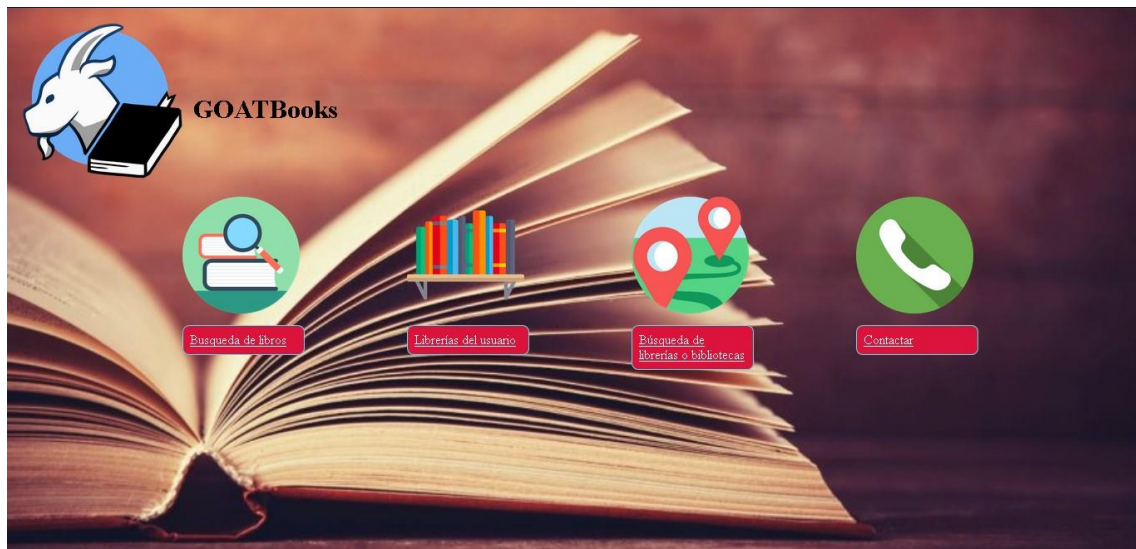
<b>ID</b>	<b>Prueba 6 – Búsqueda de información - Wikipedia</b>
<b>Descripción</b>	Prueba para la detección de errores al implementar búsquedas de información sobre autores de obras, y parseando su resultado a una lista de resultados, usando servicios RESTful.
<b>Salida esperada</b>	Los datos devueltos en formato JSON son mapeados a una clase Java y a continuación se muestra si la prueba ha sido exitosa.
<b>Resultado</b>	<b>EXITO</b>
<b>Automatizada</b>	Sí, utilizando JUnit.

- Pruebas relacionadas a Pixabay.

<b>ID</b>	<b>Prueba 7 – Búsqueda de imágenes - Pixabay</b>
<b>Descripción</b>	Prueba para la detección de errores al implementar búsquedas de imágenes sobre autores de obras, usando servicios RESTful.
<b>Salida esperada</b>	Los datos devueltos en formato JSON son mapeados a una clase Java y a continuación se muestra si la prueba ha sido exitosa.
<b>Resultado</b>	<b>Éxito parcial</b> , ya que pese a resultar correctamente, devuelve un error que no afecta a la funcionalidad de la aplicación.
<b>Automatizada</b>	Sí, utilizando JUnit.

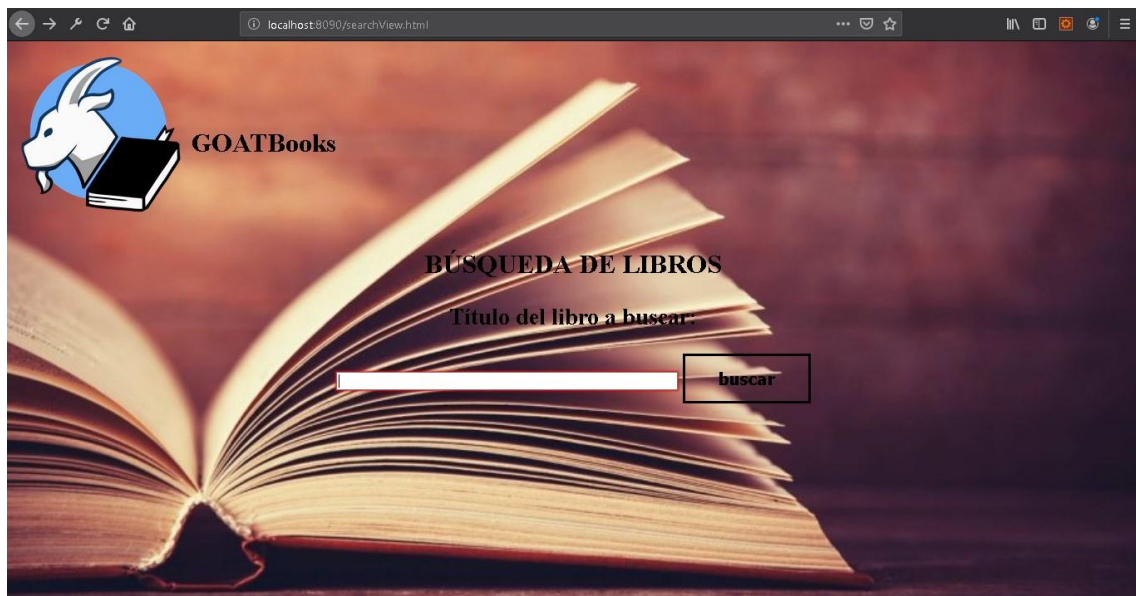
## 6 Manual de usuario

### 6.1 Mashup



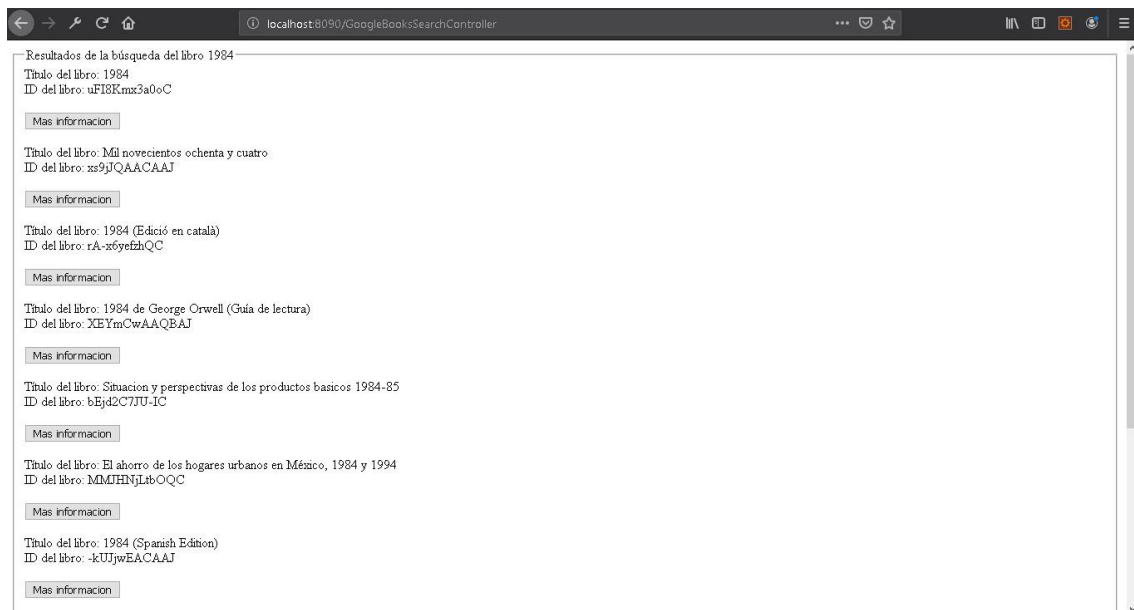
Esta es la pantalla inicial que aparece al usuario cuando accede a la aplicación, donde podemos distinguir cuatro apartados principales, en los cuales para mostrar resultados y datos se consumen las diferentes APIs.

#### 1. Búsqueda de libros.



En primer lugar, mostramos la pantalla de búsqueda, con un pequeño formulario centralizado para hacer una búsqueda de libros que tenga algún parámetro similar al introducido por el usuario.

Al introducir el título de una obra, en este caso “1984” nos mostrará los siguientes resultados:



Para cada resultado, se nos ofrece la posibilidad de consultar la información más relevante sobre la obra con pulsar el botón situado al lado del título de la obra.



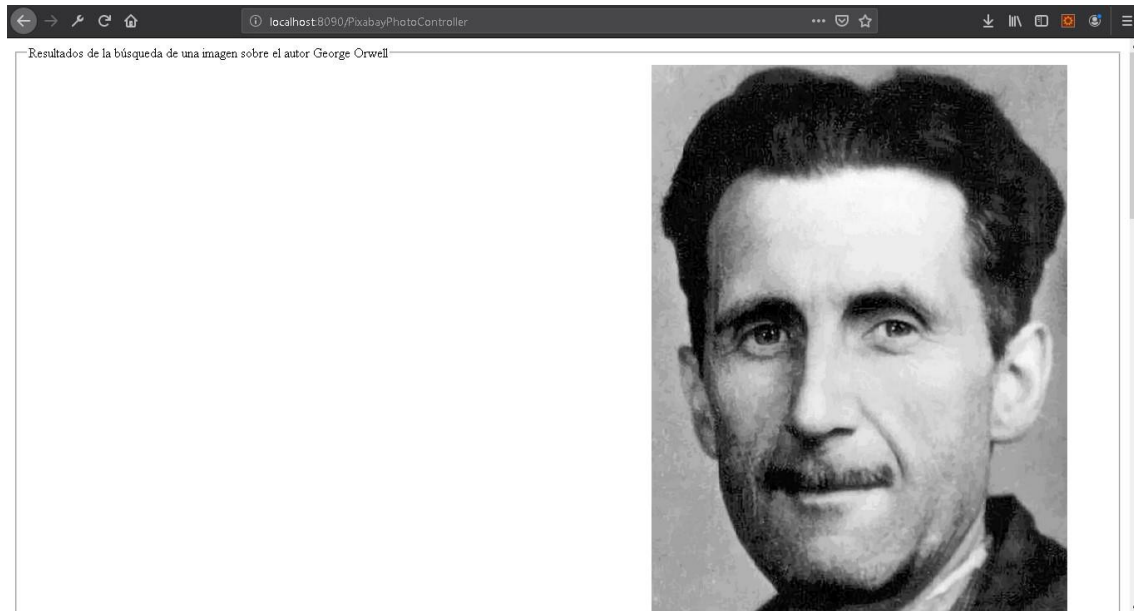
Una vez hecho click en el botón, seremos dirigidos a dicha pantalla de información.

Gracias a la integración con las librerías de Google Books, podremos añadir el libro a una librería concreta del usuario, que deberá introducir en el formulario situado al lado del botón para que realice la acción.

Si pulsamos el botón de más información del autor principal, Wikipedia nos ofrece una breve presentación con información esencial de autor.

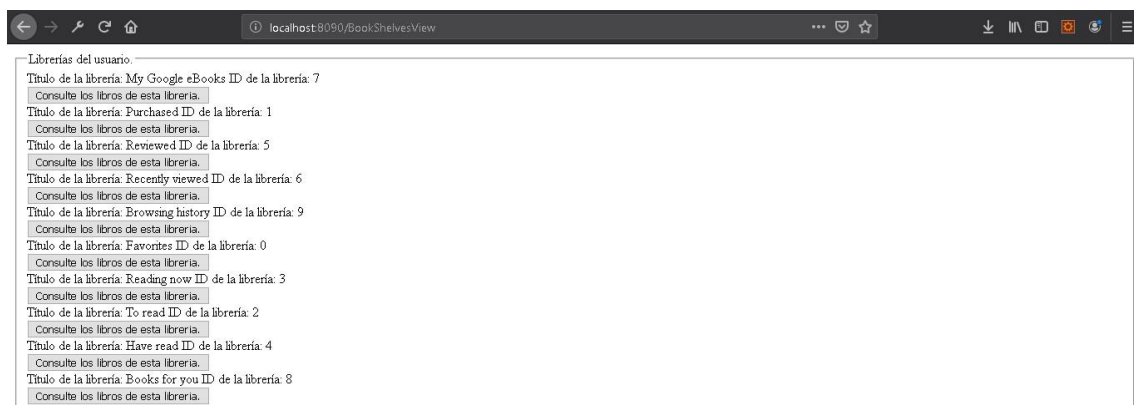


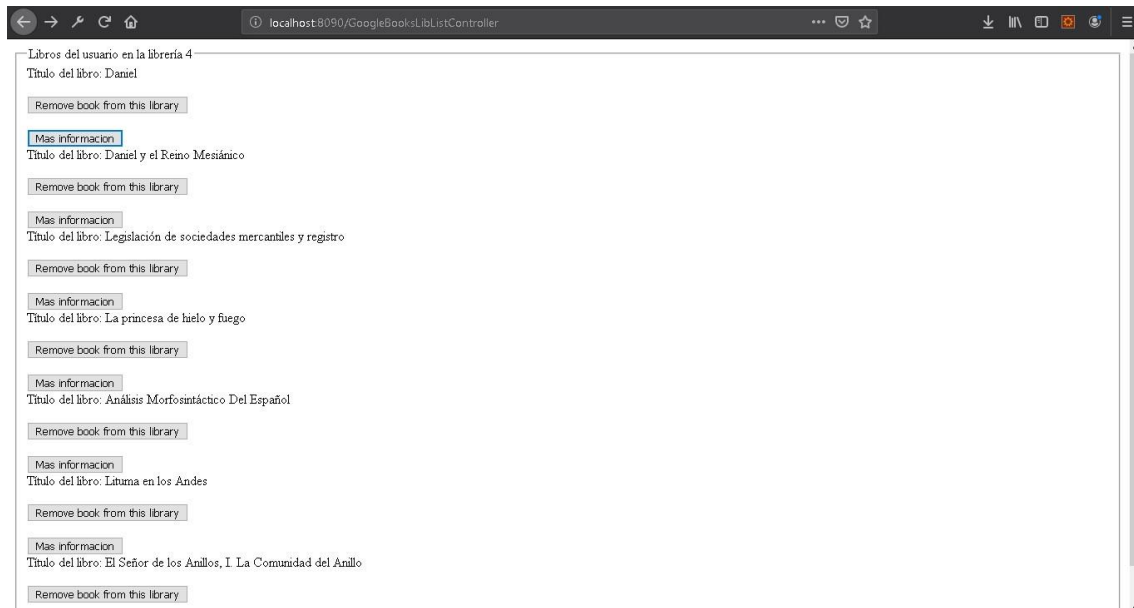
Y al presionar el botón de imagen del autor, podemos visualizar varias fotografías del mismo.



## 2. Librerías del usuario.

La segunda opción dentro de la pantalla inicial del servicio, nos muestra las librerías que tiene el usuario en Google Books, pudiendo consultar los libros que contiene cada una de ellas. Sin embargo, antes de acceder a esta vista, le será requerido autorizar el acceso de GOATBooks a la información sobre su cuenta, y concediendo los permisos necesarios para poder acceder a la misma.





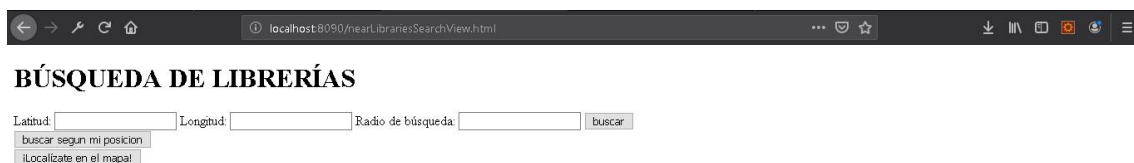
Una vez pulsado el botón de “Consulte los libros de esta librería”, se le mostrarán los libros pertenecientes a la librería que ha seleccionado, como ocurre en la vista anterior.

Una vez consultado los libros, podemos consultar la información de cada uno de los libros listados, en una pantalla como la explicada anteriormente, con la misma información y opciones.

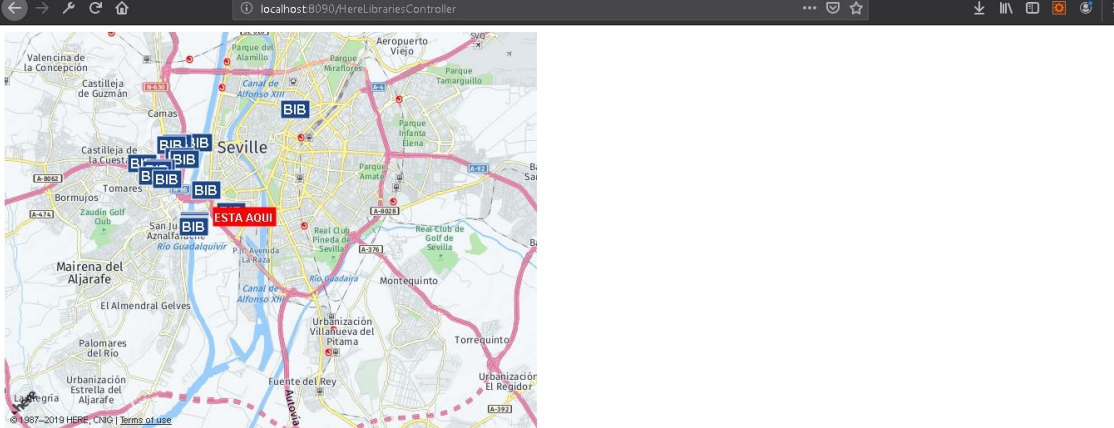
Otra opción reside en eliminar los libros contenidos que el usuario desee. Para ello, debe pulsar el botón de “Remove book from this library”.

### 3. Búsqueda de librerías.

La tercera opción en el menú principal nos permite buscar, según latitud y longitud, las librerías/bibliotecas de un radio de búsqueda que el usuario introduzca.



Por otra parte, también podemos buscar dichas librerías o bibliotecas según nuestra posición, en un radio de 10000 metros.



localhost:8080/HereLibrariesController

Resultados de la búsqueda las librerías cercanas a la posición detallada al rango 10000

Título de la librería: Librería Vertice  
Localización de la librería: [37.38171, -5.99021]  
Latitud de la librería: 37.38171  
Longitud de la librería: -5.99021

Título de la librería: Librería Reina Mercedes  
Localización de la librería: [37.361458, -5.985995]  
Latitud de la librería: 37.361458  
Longitud de la librería: -5.985995

Localización de la librería: [37.359943, -5.986061]  
Latitud de la librería: 37.359943  
Longitud de la librería: -5.986061

Título de la librería: El Cuartito - Librería y Juguetería  
Localización de la librería: [37.37144, -5.98111]  
Latitud de la librería: 37.37144  
Longitud de la librería: -5.98111

Título de la librería: Librería Guerrero  
Localización de la librería: [37.38629, -5.99572]  
Latitud de la librería: 37.38629  
Longitud de la librería: -5.99572

Título de la librería: Sport Prensa  
Localización de la librería: [37.38691, -5.98468]  
Latitud de la librería: 37.38691  
Longitud de la librería: -5.98468

Título de la librería: Librería Esteban  
Localización de la librería: [37.38682, -5.99322]  
Latitud de la librería: 37.38682  
Longitud de la librería: -5.99322

Título de la librería: Librería Botica de Lectores  
Localización de la librería: [37.3786, -6.00024]  
Latitud de la librería: 37.3786  
Longitud de la librería: -6.00024

Título de la librería: Librería Thiana  
Localización de la librería: [37.38013, -6.00765]  
Latitud de la librería: 37.38013  
Longitud de la librería: -6.00765

Su posición actual es (Latitud: 37.36111203007599, Longitud: -5.989457876515758)

[localízate en el mapa](#)

Para ver nuestra posición, presionaremos el botón de “localízate”.

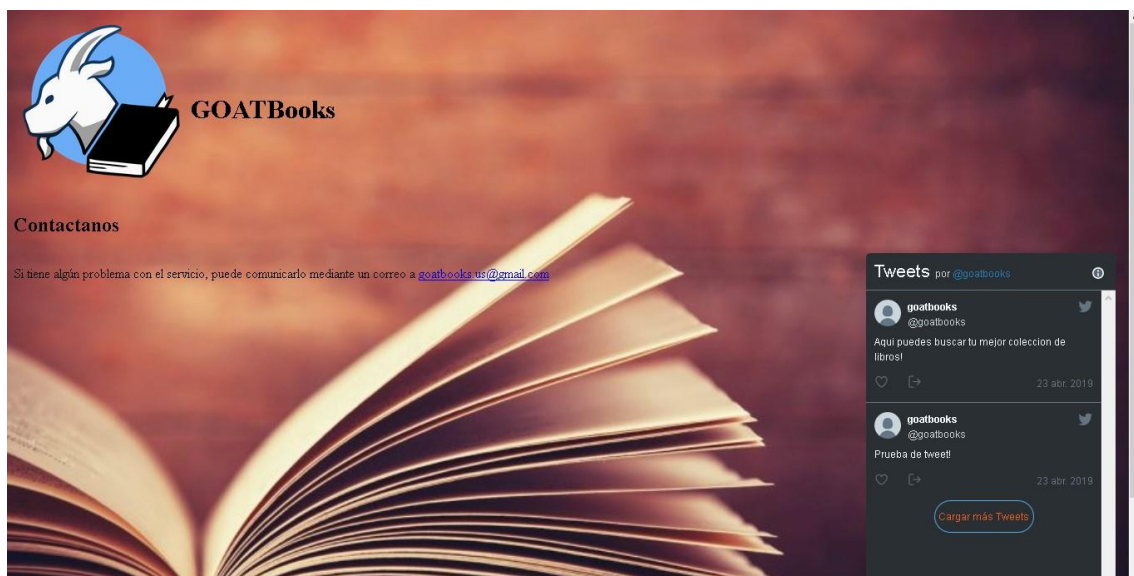
Se mostrará al usuario una lista con las diferentes librerías y bibliotecas que coincidan con la búsqueda, además de un mapa en el que también aparecerán los resultados geolocalizados en etiquetas de color azul, con el letrero “LIB”. Una vez pulsado en el botón localízate, y concedido el permiso en el navegador, también se mostrará la posición del usuario en el mapa en una etiqueta de color rojo.

#### 4. Contactar.

Se muestra la información de contacto con GOATBooks para cualquier comunicación que se le desee realizar al equipo de desarrollo y una implementación de una vista de los tweets del feed de Twitter del grupo en caso de notificaciones del servicio,



novedades y otros detalles que los mantenedores de la aplicación consideren oportuno.





## 6.2 API REST

Ofrecemos una vista documentada sobre la API en el presente documento, y le adjuntamos una URL a la documentación en formato HTML.

<https://app.swaggerhub.com/apis/goatbooks/goatbooksapi/1.0.0>

### 6.2.1 Recurso Libraries

HTTP	URI	DESCRIPCIÓN
GET	/libraries	Ver todas las librerías existentes.
GET	/libraries/{libraryId}	Devuelve la librería con id=libraryId. Si la librería no existe devuelve un "404 Not Found"
POST	/libraries	Añadir una nueva librería. Los datos de la librería (****nombre y descripción) se proporcionan en el cuerpo de la petición en formato JSON. Los libros de la librería no se pueden incluir aquí, para ello se debe usar la operación POST específica para añadir un libro a una librería. Si el nombre de la librería no es válido (nulo o vacío), o se intenta crear una librería con libros, devuelve un error "400 Bad Request". Si se añade satisfactoriamente, devuelve "201 Created" con la referencia a la URI y el contenido de la librería.
PUT	/libraries	Actualiza la librería cuyos datos se pasan en el cuerpo de la petición en formato JSON (deben incluir el id de la librería). Si la librería no existe, devuelve un "404 Not Found". Si se intenta actualizar los libros de la librería, devuelve un error "400 Bad Request". Para actualizar los libros se debe usar el recurso Song mostrado previamente. Si se realiza correctamente, devuelve "204 No Content".
DELETE	/libraries/{libraryId}	Actualiza la librería cuyos datos se pasan en el cuerpo de la petición en formato JSON (deben incluir el id de la librería). Si la librería no existe, devuelve un "404 Not Found". Si se intenta actualizar los libros de la lista, devuelve un error "400 Bad Request". Para actualizar los libros se debe usar el recurso Books. Si se realiza correctamente, devuelve "204 No Content".

<b>POST</b>	/libraries/{libraryId}/{bookId}	Añade el libro con id=songId a la librería con id=playlistId. Si la librería o el libro no existe, devuelve un “404 Not Found”. Si el libro ya está incluido en la librería devuelve un “400 Bad Request”. Si se añade satisfactoriamente, devuelve “201 Created” con la referencia a la URI y el contenido de la librería.
<b>DELETE</b>	/libraries/{libraryId}/{bookId}	Elimina la canción con id=songId de la lista con id=playlistId. Si la lista o la canción no existe, devuelve un “404 Not Found”. Si se realiza correctamente, devuelve “204 No Content”

Una librería tiene un identificador, nombre, descripción y un conjunto de libros. La representación JSON de este recurso es:

```
{
  "id": "l2",
  "name": "LibreriaFav",
  "description": "Librería donde guardo mis libros favoritos",
  "books": [
    { "id": "b4",
      "title": "Harry Potter y la piedra filosofal",
      "author": "J.K. Rowling",
      "type": "Ciencia ficcion",
      "year": "2013",
      "pages": "306",
      "lang": "esp",
      "punctuation": "9" },
    { "id": "b7",
      "title": "El alquimista",
      "author": "Paulo Coelho",
      "type": "fantasia",
      "year": "1988",
      "pages": "300",
      "lang": "esp",
      "punctuation": "3" } ]
}
```

- Representación en Swagger

libraries	
GET	/libraries Get all libraries
POST	/libraries Add a library
PUT	/libraries Update a library
GET	/libraries/{libraryId} Information about a library
DELETE	/libraries/{libraryId} Delete a library
POST	/libraries/{libraryId}/{bookId} Add book to a library
DELETE	/libraries/{libraryId}/{bookId} Delete a book from library

## 6.2.2 Recurso Books

HTTP	URI	DESCRIPCIÓN
GET	/books	Devuelve todos los libros de la aplicacion
GET	/books/{bookId}	Devuelve el libro con id=bookId; si no existe devuelve "404 Not Found".
POST	/books	Añade un nuevo libro cuyos datos se pasan en el cuerpo de la petición en formato JSON (no se debe pasar id, se genera automáticamente). Si el nombre del libro no es válido (null o vacío) devuelve un error "400 Bad Request". Si se añade exitosamente, devuelve "201 Created" con la referencia a la URI y el contenido del libro.
PUT	/books	Actualiza el libro cuyos datos se pasan en el cuerpo de la petición en formato JSON (deben incluir el del libro). Si el libro no existe, devuelve un "404 Not Found". Si se realiza correctamente, devuelve "204 No Content"
DELETE	/books/{bookId}	Elimina el libro con id=bookId. Si la canción no existe, devuelve un "404 Not Found". Si se realiza correctamente, devuelve "204 No Content".

Cada libro tiene un identificador, título, nombre del autor, género, año de publicación, número de páginas, idioma y una valoración. La representación JSON del recurso es:

```
{
  "id": "b4",
  "title": " Harry Potter y la piedra filosofal ",
  "author": " J.K. Rowling ",
  " type ": "Ciencia ficcion",
  " year ": "2007"
  "pages": "306",
  "lang": "esp",
  "punctuation": "9"
}
```

- Representación en Swagger

books		▼
GET	/books	Get all books
POST	/books	Create a book
PUT	/books	Update a book
GET	/books/{bookId}	Information about a library
DELETE	/books/{bookId}	Delete a book

### 6.2.3 Recurso BookShops

HTTP	URI	DESCRIPCIÓN
GET	/bookShops	Devuelve todas las bibliotecas cercanas de la aplicación.
GET	/bookShops/{bookShopId}	Devuelve la biblioteca con id= bookShopId; si no existe devuelve "404 Not Found".

Cada biblioteca tiene un identificador, nombre, localización, horario de apertura y horario de cierre. La representación JSON del recurso es:

```
{
  "id": "b2",
  "name": "Biblioteca CRAI de la Universidad Pablo de Olavide de Sevilla",
  "location": "Sevilla",
  "open": "15:30",
  "close": "20:30"
}
```

- Representación en Swagger

bookshops		▼
GET	/bookShops	Get all book shops
GET	/bookShops/{bookShopId}	Information about a book shop

## Referencias

- [1] *Balsamiq*. <http://balsamiq.com/>. Accedido en Enero 2014.
- [2] J. Webber, S. Parastatidis y I. Robinson. *REST in Practice: Hypermedia and Systems Architecture*. O'Reilly Media. 2010.