E.T.S. de Ingeniería Industrial,
Informática y de Telecomunicación

Diseño y desarrollo de una herramienta web de seguimiento de ventas de libros electrónicos

Grado en Ingeniería Informática

Trabajo Fin de Grado

Rubén San Miguel Herrera

José Javier Astrain Escola

Pamplona, 27-06-2014

upna Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

RESUMEN

El objeto de este TFG es el desarrollo de una aplicación para la empresa Leer-e, que comercializa libros electrónicos a través de diferentes plataformas (Leer-e, Amazon, Apple, Google...).

Estas plataformas generan unos informes mensualmente (en formato de hoja de cálculo) que contienen información de seguimiento de las ventas de sus libros en cada una de ellas. Esta información es similar para todas las plataformas, pero no igual, y a día de hoy es tratada manualmente para ofrecer un seguimiento de las ventas.

La elaboración de este TFG, consiste en el diseño y desarrollo de una aplicación web, en la cual se van a uniformizar y automatizar estas tareas. Esta aplicación se ha realizado con Symfony2, mediante los lenguajes de programación siguientes:

- PHP
- HTML
- CSS
- DOCTRINE
- TWIG
- JS

Para concluir, este proyecto, se ha llevado a cabo en las propias instalaciones de la empresa Leer-e y bajo su supervisión.

PALABRAS CLAVE

Antes de comenzar a dar detalles sobre este TFG, voy a definir las palabras clave las cuales se van a utilizar a lo largo de la memoria. Estas palabras son:

BBDD: Acrónimo de base de datos.

Función/Funciones: A lo largo de la memoria se mencionan estas palabras para hablar del rol que tiene un usuario en cuanto a un libro, puden ser: escritor, traductor, ilustrador, agente y director.

Royalty: Porcentaje que gana la empresa o los usuarios sobre un libro.

Tipo de cambio: Número de unidades de la moneda nacional que hay que entregar, en un momento dado, a cambio de una unidad de moneda extranjera.

Moneda: Tipo de moneda en el que se efectúa la venta de los libros: euros, dolares…

Modelo: Parte de una aplicación que se encarga de gestionar la comunicación con la bbdd.

Vista: Parte de una aplicación quehace referencia a la interfaz de la aplicación.

Controlador: Parte de una aplicación que hace referencia a lo que controla las funciones de una aplicación.

JS: Abreviatura del lenguaje JavaScript.

Clase: Las clases se utilizan para representar entidades o conceptos.

Método: Subrutina cuyo código es definido en una clase.

Framework: Es una estructura conceptual y tecnológica de soporte definido, normalmente con módulos de software concretos, que puede servir de base para la organización y desarrollo de software. Puede incluir soporte de programas, bibliotecas… para así ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Rol: Función o papel que desempeñan los distintos usuarios de la aplicación.

Agente: Representante de autores.

Plataforma: Distintas editoriales con las que trabaja Leer-e.

Path: Ruta almacenada en la bbdd, para almacenar imágenes, archivos…

Salt: Campo en las entidades de usuarios para almacenar la contraseña cifrada.

Interfaz: parte de un programa que permite el flujo de información entre un usuario y la aplicación.

Backend: Parte de la aplicación referente a los administradores.

Frontend: Parte de la aplicación referente a los usuarios.

Cifrar: proteger archivos expresando su contenido en un lenguaje cifrado.

INDICE

1.- INTRODUCCIÓN

En este capítulo se dará información sobre la Empresa, se explicará el problema a resolver con sus objetivos y la solución propuesta, además se detallarán las tecnologías utilizadas.

1.1 Empresa LEER-E

1.1.1 Información sobre la empresa

////////////////////////////////////

Esta empresa, antiguamente, se dedicaba a la venta ebooks, pero dejó de trabajar en ello hace bastantes años, ya que se dieron cuenta de que no aportaba suficiente beneficio y acarreaba multitud de problemas en cuanto a garantías, reparaciones y demás.

Pasado un tiempo, observaron que lo que en realidad era interesante, era la venta de los libros en formato electrónico. Esta decisión les repercutió mucho éxito, tanto que a día de hoy, Leer-e, es una de las tiendas, de las pocas que se dedican a este negocio, que funciona muy bien y está teniendo bastante éxito.

www.leer-e.es [1] es la página web de la empresa, donde se puede acceder para ver y comprar con total seguridad los libros que el usuario desee. Además muestra ayudas en el caso de que un usuario no domine la informática y quiera aprender sobre qué pasos realizar para poder leer su libro, etc.

Aparte de todo lo anterior, Leer-e, vende sus libros en otras plataformas, como Apple, Amazon, Google, ya que creen que son interesantes para poder vender en el extranjero por ejemplo, y son muy conocidas por los usuarios.

Estas plataformas, se comprometen a realizar unos informes mensuales sobre el seguimiento de las ventas de los libros, los cuales son independientes unos de otros.

Estos informes son similares, pero no iguales, por lo que la información necesita ser tratada antes de ser examinada y almacenada.

Una vez que la información ha sido tratada por un empleado de Leer-e, esta es transmitida a cada representante del libro, y así ellos puedan ver el éxito de sus libros y lo que van a recibir por sus ventas. Estos representantes de dividen en:

- Autores: escritor, diseñador, traductor del libro.
- Agentes: son personas que representan a un Autor o a varios.
- Directores: son Autores o Agentes, los cuales cobran un plus por ello.

1.1.2 Datos de interés

En este apartado, voy a intentar ubicar al lector en cuanto a la jerarquía de palabras que se utilizan en este tipo de sector. Para que cuando aparezcan a lo largo de este documento, se sepa de que se está hablando.

En mi caso, cuando llegué a la empresa, me sonaba todo ‘raro’, ya que me empezaban a hablar de cosas que no sabía a qué hacían referencia, y hasta que no vas trabajando con ello no sabes muy bien de qué están hablando.

Ahora voy a describir algunos aspectos para intentar situar al lector:

En primer lugar, en la aplicación sólo existe una tabla de usuarios, pero estos tendrán libros en diferentes tablas según su rol. En la relación usuarios-libros, existen cinco relaciones, una tabla por cada tipo de rol, que son las siguientes:

- Escritores-libros
- Ilustradores-libros
- Traductores-libros
- Directores-libros
- Agentes-libros

Dependiendo de si un autor tiene libros en cada una de estas relaciones, a la hora de vender sus libros, obtendrá un porcentaje u otro. Es decir:

- Si un usuario posee un libro, siendo escritor y director de este, obtendrá el porcentaje perteneciente por ser escritor, y el porcentaje correspondiente por ser director.

Esto se reflejará en los informes que genera la aplicación automáticamente, donde se separan en hojas los libros en correlación con el rol que tengan estos.

Seguido a esto, un libro puede tener muchas editoriales y géneros, es un caso raro pero podría darse el caso, pero sólo puede tener una colección.

Por otro lado, existe un campo llamado Royalties. Este campo viene en los informes que obtenemos de las distintas plataformas, y su significado viene a ser el porcentaje del precio que le corresponde a la editorial Leer-e por la venta de cada libro. Este atributo también existirá para cada usuario, donde a la hora de generar los informes de las ventas, según el rol que desempeñe este sobre el libro, le corresponderá un porcentaje u otro, este porcentaje es lo que llamamos Royalties. Este dato estará almacenado en la tabla libros.

El royalty de la empresa Leer-e hace referencia al precio de la venta en cada una de las plataformas, mientras que el royalties de los usuarios, es un porcentaje que hace referencia al precio del libro. Esto no es lo mismo, ya que el autor tiene el libro a un precio y ellos lo venden a otro.

Además a la hora de obtener las ventas de los libros, existe en estos ficheros un campo llamado: ‘tipo de cambio’, este campo se utiliza para hacer una conversión de las distintas monedas extranjeras al euro. En conclusión, las ventas almacenadas en mi base de datos se guardan todas en formato euro, realizando una conversión a la hora de leer los ficheros. Esta conversión puede ser automática al instante de insertar la venta ya que en el fichero existen los datos necesarios, o se puede hacer de forma manual. Esta segunda forma tuvo que implementarse porque determinadas plataformas (BN y Apple) pasan estos valores al cabo de los dos meses, porque según países, estos valores van cambiando, con lo cual es un poco más engorroso pero es necesario. Se realizará de dos formas las cuales detallaré más adelante.

1.2 Problema a resolver

La propuesta de este proyecto la transmitió la empresa ya citada, Leer-e, la cual se puso en contacto con la Universidad Pública de Navarra, para dar la oportunidad a un estudiante de realizar las prácticas en ella, realizando el trabajo en sus instalaciones, lo cual supondrá el trabajo fin de grado del estudiante.

La idea del proyecto ofertado, surgió de la necesidad de uniformizar y automatizar las tareas antes comentadas, ya que a día de hoy es una pérdida de tiempo y dinero para esta empresa.

En primer lugar, hay que destacar:

- Falta de una base de datos donde estén almacenados todos los registros de sus ventas.
- Falta de una base de datos, donde tengan la información de todos los libros
- Falta de una base de datos, donde tengan la información de todos los autores.
- Falta de una base de datos, donde tengan la información de todos los agentes.

Lo cual tiene varios hándicaps:

- Para poder hacer un seguimiento de lo que van vendiendo, tienen que buscar entre todos las hojas de cálculo para ver lo que deseen.
- Perdida de dinero al tener a un empleado uniformizando, todos los meses, los informes obtenidos de cada plataforma.
- Pérdida de tiempo, al tener que enviar a todos los autores y agentes sus respectivos informes.

Por todos los motivos citados anteriormente y por todo lo que conlleva, vieron la necesidad de implementar esta aplicación web, ya que internet, es la herramienta más utilizada a día de hoy por personas de todas las edades.

La intención es realizar una aplicación de uso intuitivo, para que cualquier persona, ya sea joven, de mediana edad o personas mayores, pueda utilizar sin ningún tipo de problema esta aplicación. Ya que existen autores de mayor edad los cuales no están familiarizados a usar herramientas de este tipo.

Los usuarios podrán generar sus propios informes cuando lo deseen, ya sea por mes, trimestral, anual, etc. Además tendrán la opción de hacer un seguimiento a simple vista con la ayuda de gráficas generadas en tiempo real, donde seleccionarán el periodo y determinados filtros y podrán ver cómo van sus ventas.

La aplicación estará accesible públicamente en un dominio de la empresa, adonde los usuarios podrán acceder libremente cuando lo deseen, y allí realizar sus operaciones.

Por otro lado, esta aplicación posee un backend, donde los administradores se encargarán de administrar la base de datos, realizando operaciones como, alta baja y modificación de datos. Además tendrán un apartado en el cual accederán para subir los ficheros que las diferentes plataformas les facilitan todos los meses. Estos sólo tendrán que subir los ficheros, sin ninguna operación más, ya que a la hora de insertarse serán tratados según pertenezca a una plataforma u a otra, almacenándose correctamente en la base de datos.

Por último, la propia empresa me dijo con qué herramienta de desarrollo tenía que realizar la aplicación. Debido a que el informático de allí la dominaba bastante, esta era Symfony [2], un framework de php [3] el cual no sabía utilizar. Esta decisión por su parte se debió a que es una herramienta la cual te facilita mucho las cosas a la hora de realizar una aplicación web.

1.3 Objetivos

Durante la implementación de esta aplicación, he tenido en cuenta diferentes objetivos. Por un lado, una vez elegí realizar este proyecto, no sabía muy bien a lo que me iba a enfrentar, ya que la información que me dieron sobre él era mínima. Una vez supe cuáles eran los objetivos me replanteé los míos propios.

- Aprender a desarrollar proyectos con la herramienta Symfony.
- Conocer la estructura en carpetas de cómo trabaja esta herramienta.
- Profundizar mis conocimientos de programación de los siguientes lenguajes:
  - Php [3]
  - Html [4]
  - Css [5]
  - JavaScript [6]
- Aprender a elaborar y administrar bases de datos con Doctrine [7].
- Aprender a utilizar las plantillas de Twig [8], ya que es con lo aconseja trabajar Symfony, siendo estas las vistas de la aplicación.
- Volver a recordar mis conocimientos sobre las bases de datos (consultas…).
- Aprender a trabajar en un entorno de trabajo.
- Comprometerme en realizar correctamente mis funciones.
- Satisfacer las necesidades de la empresa, y a la vez, satisfacerme a mí mismo.

Por otro lado, los objetivos de la aplicación que he realizado son bastante claros.

- Almacenar toda la información de las ventas, usuarios, libros en una base de datos.
- Creación de un sitio web para el seguimiento de las ventas de los clientes.
- Agilizar el proceso de elaboración de informes a los clientes.
- Apartado en la web, donde los clientes puedan ver a simple vista, mediante gráficas, la información de sus ventas.
- Ahorrar trabajo a la empresa, automatizando este proceso.

Todos estos objetivos, son los que se pretende conseguir implementando la aplicación anteriormente descrita.

Solo el mero hecho de tener los datos almacenados en una base de datos, es un paso gigante que se va a conseguir, ya que a día de hoy, si no tienes los datos almacenados de esta forma, el día de mañana será una auténtica locura.

Además, pensando en los clientes, les va a resultar muy cómodo el tener los informes cuando ellos deseen.

1.4 Estado del arte

En este apartado, voy a hablar de las herramientas tecnológicas que he utilizado para el desarrollo de mi aplicación, en especial Symfony.

Representa un proyecto PHP de software libre que permite crear de forma profesional, aplicaciones y sitios web seguros de manera rápida.

Este framework está basado en el patrón Modelo Vista Controlador, ya que fue diseñado para optimizar el desarrollo de aplicaciones web. Debido a que este patrón es a día de hoy el más utilizado por la forma en que estructura la información. Para empezar:

- Separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web.
- Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja.
- Automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación.

El objetivo de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación web. De ahí surgió esta idea.

Fue desarrollado en PHP 5.3 y ha sido probado en multitud de proyectos reales con muy buenos resultado. Una de las características más importantes, es que Symfony, es compatible con la mayoría de gestores de bases de datos, como MySQL [9], PostgreSQL [10], Oracle [11] y Microsoft SQL Server [12]. Hay que añadir que se puede ejecutar tanto en plataformas *nix (Unix, Linux, etc.) como en plataformas Windows.

Características más significativas:

- Fácil de instalar y configurar en la mayoría de plataformas.
- Independiente del sistema gestor de bases de datos.
- Utiliza programación orientada a objetos.
- Uso sencillo.
- Utiliza MVC (Modelo Vista Controlador.
- Fácil de extender, lo que permite su integración con las bibliotecas de otros fabricantes.
- Una potente línea de comandos que facilitan generación de código, lo cual contribuye a ahorrar tiempo de trabajo.
- Permite la internacionalización para la traducción del texto de la interfaz, los datos y el contenido de localización.
- Los formularios soportan la validación automática, lo cual asegura mejor calidad de los datos en las base de datos y una mejor experiencia para el usuario.

- El manejo de cache reduce el uso de banda ancha y la carga del servidor.
- La facilidad de soportar autenticación y credenciales.

La Figura 1.1 refleja las distintas versiones que se han ido generando con el paso de los años, añadiendo mejoras y funciones nuevas.

Versión	Fecha lanzamiento	Soporte	Versión de PHP	Fin de mantenimiento	Notas		
1.0	Enero 2007	3 años	>= 5.0	Enero 2010		Color	Meaning
1.1	Junio 2008	1 año	>= 5.1	Junio 2009	parches de seguridad	Rojo	Versión ya no soportada
1.2	Diciembre 2008	1 año	>= 5.2	Noviembre 2009		Verde	Versión actualmente soportada
1.3	Noviembre 2009	1 año	>= 5.2.4	Noviembre 2010		Azul	Versión futura
1.4	Noviembre 2009	3 años	>= 5.2.4	Noviembre 2012	Versión LTS 1.4 es idéntica a 1.3, pero no es compatible con las funcionalidades obsoletas de 1.3.[7]		
2.0[8]	Julio 2011[9]		>= 5.3.2	Marzo 2013	La última versión 2.0.x fue Symfony 2.0.23[10]		
2.1[11]	Septiembre 2012	8 meses	>= 5.3.3	Abril 2013	Más componentes son parte del API estable.		
2.2	Marzo 2013	8 meses	>= 5.3.3	Noviembre 2013	Nuevas características[12]		
2.3	Mayo 2013	3 años	>= 5.3.3	Mayo 2016	(la primera versión LTS) fue lanzada a finales de mayo de 2013;		
2.4	Noviembre 2013	8 meses	>= 5.3.3	Julio 2014	La primera versión de la rama 2.x que tiene retrocompatibilidad completa.[13]		

Figura 1.1 Versiones de Symfony. Fuente: Wikipedia, 2014.

Symfony para programadores:

Definición: Symfony es un framework PHP de tipo full-stack construido con varios componentes independientes creados por el proyecto Symfony.

Principales características:

- Su código, y el de todos los componentes y librerías que incluye, se publican bajo la licencia MIT de software libre.
- La documentación del proyecto también es libre e incluye varios libros y decenas de tutoriales específicos.
- Aprender a programar con Symfony te permite acceder a una gran variedad de proyectos: el framework Symfony2 para crear aplicaciones complejas, el micro framework Silex para sitios web sencillos y los componentes Symfony para otras aplicaciones PHP.
- Los componentes de Symfony son tan útiles y están tan probados, que proyectos tan gigantescos como Drupal 8 están construidos con ellos.
- En todo el mundo se celebran varias conferencias dedicadas exclusivamente a Symfony.

Symfony para administradores de sistemas:

Definición: Symfony es un conjunto de librerías que se utilizan para crear aplicaciones PHP.

Principales características:

- Las versiones actuales de Symfony requieren disponer de PHP 5.3.8 o superior. Así evitas instalar en tus servidores versiones PHP peligrosas llenas de problemas de seguridad y a la vez no es un requisito técnico demasiado exigente.
- En producción, las aplicaciones Symfony solamente necesitan permiso de escritura en dos directorios internos de la propia aplicación. Además, Symfony incluye varias herramientas gráficas y de consola para depurar fácilmente los errores que se produzcan en las aplicaciones.
- Para evitar el uso de contraseñas en archivos de configuración, Symfony permite establecer los parámetros de configuración de las aplicaciones a través de variables de entorno del propio servidor.
- La seguridad es tan importante para el proyecto Symfony, que antes de su lanzamiento, se encargó una auditoría de seguridad a una empresa independiente.
- La excelente herramienta Composer, que simplifica de forma radical la instalación y gestión de las dependencias de las aplicaciones PHP, también ha sido creada por varios miembros de la comunidad Symfony.

PHP

PHP (acrónimo recursivo de *PHP: Hypertext Preprocessor*) es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML.

El código de PHP está encerrado entre las etiquetas especiales de comienzo y final <?php y ?> que permiten entrar y salir del "modo PHP". Lo que distingue a PHP de algo como Javascript del lado del cliente es que el código es ejecutado en el servidor, generando HTML y enviándolo al cliente. El cliente recibirá el resultado de ejecutar el script, aunque no se sabría el código subyacente que era. El servidor web puede ser incluso configurado para que procese todos los ficheros HTML con PHP, por lo que no hay manera de que los usuarios puedan saber qué se tiene debajo de la manga.

Existen principalmente tres campos principales donde se usan scripts de PHP.

- Scripts del lado del servidor.
- Scripts desde la línea de comandos
- Escribir aplicaciones de escritorio

Características:

- Puede usarse en todos los principales sistemas operativos, incluyendo Linux, muchas variantes de Unix (incluyendo HP-UX, Solaris y OpenBSD), Microsoft Windows...
- PHP admite la mayoría de servidores web de hoy en día, incluyendo Apache, IIS, y muchos otros.
- Entre las capacidades de PHP se incluyen la creación de imágenes, ficheros PDF e incluso películas Flash (usando libswf y Ming) generadas sobre la marcha.
- También se puede generar fácilmente cualquier tipo de texto, como XHTML y cualquier otro tipo de fichero XML.
- Una de las características más potentes y destacables de PHP es su soporte para un amplio abanico de bases de datos.
- También cuenta con soporte para comunicarse con otros servicios usando protocolos tales como LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM (en Windows) y muchos otros.

CSS

Hojas de Estilo en Cascada (Cascading Style Sheets) es el lenguaje de hojas de estilo utilizado para describir el aspecto y el formato de un documento escrito en un lenguaje de marcas, como puede ser HTML.

Estas reglas las cuales definen el estilo de un documento, pueden definirse en diferentes sitios:

1. Un estilo en línea (online) es un método para insertar el lenguaje de estilo de página directamente dentro de una etiqueta HTML.
2. Una hoja de estilo interna, que es una hoja de estilo que está incrustada dentro de un documento HTML, dentro del elemento <head>, marcada por la etiqueta <style>.
3. Una hoja de estilo externa, es una hoja de estilo que está almacenada en un archivo diferente al archivo donde se almacena el código HTML de la página Web.

Una hoja de estilo se compone de una lista de reglas. Cada regla o conjunto de reglas consiste en uno o más selectores y un bloque de declaración (o «bloque de estilo») con los estilos a aplicar para los elementos del documento que cumplan con el selector que les precede. Cada bloque de estilos se define entre llaves, y está formado por una o varias declaraciones de estilo con el formato (propiedad:valor;)

JavaScript

Es un lenguaje de programación interpretado, muchas veces nos lo vamos a encontrar escrito de la siguiente forma ‘JS’. Se define como orientado a objetos, basado en prototipos, débilmente tipado y dinámico.

Se utiliza principalmente para realizar aplicaciones web, donde normalmente es aplicado al lado del cliente. De esta forma permite añadir mejoras en la interfaz del usuario, realizando los sitios más atractivos y llamando la atención del usuario.

Como he dicho, JS, es un lenguaje interpretado, esto quiere decir que no es necesario compilar el código para poder ejecutar la aplicación. Una de sus características más importantes es que ves en todo momento lo que estás haciendo, es decir, una vez realizado un cambio en tu código, lo ves al instante.

Doctrine

Doctrine es un mapeador de objetos-relacional (ORM) escrito en PHP que proporciona una capa de persistencia para objetos PHP. Es una capa de abstracción que se sitúa justo encima de un SGBD.

Por ejemplo, si un programador quiere crear un nuevo objeto "Usuario" en la base de datos, no tendrá que escribir ninguna sentencia SQL, simplemente lo siguiente:

```
$user = new User();
$user->name = 'Juan';
$user->password = '123';
$user->save();
```

Características:

- Doctrine puede generar clases a partir de una base de datos existente y después el programador puede especificar y añadir funcionalidad extra a las clases autogeneradas.
- Posibilidad de escribir consultas de base de datos utilizando un dialecto de SQL denominado DQL (Doctrine Query Language).
- Herencia

Plantillas Twig

Symfony2 incluye un lenguaje de plantillas llamado Twig [8] que es mucho más potente y elegante que PHP. Gracias a Twig puedes crear plantillas muy concisas y fáciles de leer, por lo que además son fáciles de entender por parte de los diseñadores web.

De esta forma, al utilizar una plantilla general, en cualquier momento puedes cambiarla y no tendrás que realizar ningún cambio más en el resto de las plantillas.

La sintaxis de Twig se basa en dos etiquetas especiales:

- {{ ... }}: sirve para mostrar el contenido de una variable o el resultado de realizar alguna operación o procesar alguna expresión. En PHP la construcción equivalente es echo o print.
- {% ... %}: sirve para definir la lógica de la plantilla, es decir, la parte de programación que controla cómo se muestran los contenidos de la plantilla. Entre otros, esta etiqueta se emplea para las instrucciones if y para los bucles for.
- {# … #}: sirve para comentar código.

Otros datos de interés:

Para tener una ligera idea de lo que pretendían conseguir, observé las páginas de las diferentes plataformas donde ellos acceden con su usuario para poder descargar los informes mensuales de sus ventas. El objetivo final siempre es el mismo, marcar la fecha de la cual quieres recibir el informe y automáticamente te da la opción de descargarlo en el formato que tiene asignado.

Otras páginas sacan un listado con los informes nuevos del mes, y directamente el usuario se los descarga. Pero estudiando las necesidades de nuestros clientes, optamos por la opción de elegir fechas y crear informes, ya que de esta forma los usuarios podrán generar informes trimestralmente y anualmente si lo desean.

Por otro lado, para determinar cómo mostrar la información en gráficas, observé la página de Apple. Apple, internamente, muestra a sus clientes el seguimiento de las ventas mediante gráficas. Esta plataforma interactúa con multitud de filtros aplicables, los cuales en nuestro caso son innecesarios.

Nuestra idea se basa en un breve resumen de las ventas mensuales y anuales de todos los libros de cada autor o agente. También posee la opción de elegir un libro, y hacer un seguimiento de sus ventas, como por ejemplo, ver mes a mes lo que se ha vendido de este.

1.5 Solución propuesta

Después de analizar todo lo anterior, mi propuesta es realizar una aplicación sencilla e intuitiva de utilizar. Estará implementada con la herramienta Symfony2, y los lenguajes utilizados serán: PHP, HTML, CSS y JS en la mayor parte del código. Aparecerá un menú básico donde los usuarios podrán moverse sin dificultad.

Facilidad de ver el progreso de sus ventas mediante gráficas, para que en cualquier momento puedan ver el progreso de sus libros.

Daré la opción a los administradores, de administrar la base de datos, bien sea de libros, usuarios o de ventas.

Esta aplicación será alojada en un dominio reservado por la empresa una vez estén todos los datos asentados. Acto seguido se les informará a los clientes de Leer-e de la existencia de esta, y se les proporcionará un usuario y contraseña.

Tendrán acceso desde cualquier máquina, ya sea Windows, Linux o Mac, además al ser una aplicación web, se podrá acceder desde cualquier dispositivo móvil o tablet, ya que la interfaz se adaptará al tamaño de la pantalla.

2.- ANALISIS DE REQUISITOS

En este capítulo se explicarán los requisitos de la aplicación y se mostrarán los diagramas de casos de uso.

En este punto, se van a detallar los requisitos de la aplicación, tanto funcionales como los no funcionales, realizando un análisis de la aplicación web.
La principal idea es mostrar una interfaz sencilla, la cual representará a la vista, donde el controlador recogerá las órdenes del usuario, y este será el encargado de interactuar con la base de datos (el modelo), para así mostrarles en cada momento la información que ellos necesiten.

Todos los requisitos nombrados a continuación se han concretado después de varias reuniones con él con el operario de la empresa el cual se encargaba de realizar estos informes, ya que es la persona que más ha trabajado con esto y la que más conoce las características de cada informe, plataforma clientes y demás.

Esta aplicación es de carácter informativo, lo cual requiere los siguientes requisitos:

2.1.1 Requisitos funcionales:

- Login para el usuario.
- Menú visible en todo momento, con los diferentes submenús integrados en cada apartado.
- Dirección de la empresa, mediante Google Maps.
- Acceso directo a las diferentes páginas de cada plataforma.
- Opción de descargar informe detallado del periodo seleccionado para las ventas de cada usuario.
- Opción de visualizar mediante gráficas, un resumen detallado se sus ventas, para visualizar a simple vista, el progreso de las mismas.
- Información de la empresa Leer-e.
- Breve explicación para saber manejar la aplicación en el caso de que se tengan dudas.
- Opción de visualizar los detalles de sus libros, así como portada, género…
- Login para el administrador.
- Funciones para que el administrador pueda administrar la base de datos, insertar, borrar, actualizar…
- Inserción de las ventas en la base de datos mediante la subida de un fichero, lo cual supone un ahorro de tiempo y trabajo para la empresa.
- Opción múltiple para subir ficheros de la misma plataforma, debido a que las plataformas grandes, generan un informe por país. De esta forma se podrán seleccionar todos a la vez y realizar la inserción.
- Opción de actualizar las ventas, ya que Apple y BN proporcionan el valor del tipo de cambio de la moneda el mes siguiente, con lo cual hay que actualizar las ventas insertadas anteriormente, mediante una cantidad, o un fichero.
- Comprobación automática antes de insertar una venta de su existencia, ya que de vez en cuando, a las plataformas se les cuela este tipo de problema.
- Opción de eliminar el informe completo de una venta, debido a que puede darse el caso de que una plataforma después de generarte el informe, avise a los días de que ese archivo está mal y que recomiende la descarga de uno

nuevo. Esto implica la eliminación de los datos que ya se han insertado anteriormente.
- Seguridad a la hora de ingresar en la aplicación mediante algoritmo de encriptación SHA512.
- Un buscador para buscar por autor, libros o lo que se desee.

2.1.2 Requisitos no funcionales:

- La aplicación, va a manejar una gran cantidad de datos, con lo cual esta información introducida debe ser la mínima y necesaria para agilizar el proceso de mostrar información.
- Proporcionar una interfaz intuitiva y sencilla, donde cualquier persona sea capaz de realizar las consultas deseadas sin problemas.
- La aplicación no debe dar errores, y si los da, se mostrará una página de error acorde con la aplicación.
- Tipo de letra legible, nada de letras extravagantes.
- Opción de volver atrás en cualquier momento.
- Opción de hacer logout cuando se desee.
- Diseño adaptable para cualquier tamaño de pantalla.

2.2 Diagramas de caso de uso

Los casos de uso describen en forma de acciones y reacciones el comportamiento de un sistema desde el punto de vista del usuario, ya que permiten definir los límites y las relaciones entre este y el entorno. Además, son descripciones de la funcionalidad del sistema independientes de cómo se realice la implementación.

Los actores de estos se determinan observando lo siguiente:

- Usuarios que interaccionan con el sistema.
- Responsables del uso o mantenimiento del sistema.
- Otros sistemas que interactúen con el sistema de la aplicación.

Los nombres que definen estos actores, viene a definir el papel que desempeñan en la aplicación, además hay que añadir que una misma persona física puede desempeñar varios papeles como actores diferentes.

En este apartado voy a mostrar los diferentes diagramas de casos de uso correspondientes a mi aplicación. Por un lado se mostrarán los casos de uso de *frontend* (correspondientes a la parte de los usuarios), y por otro los del *backend* (correspondientes a la parte de administración).

2.2.1 Casos de uso del frontend:

2.2.1.1 Diagrama de Caso de Uso para el Login.

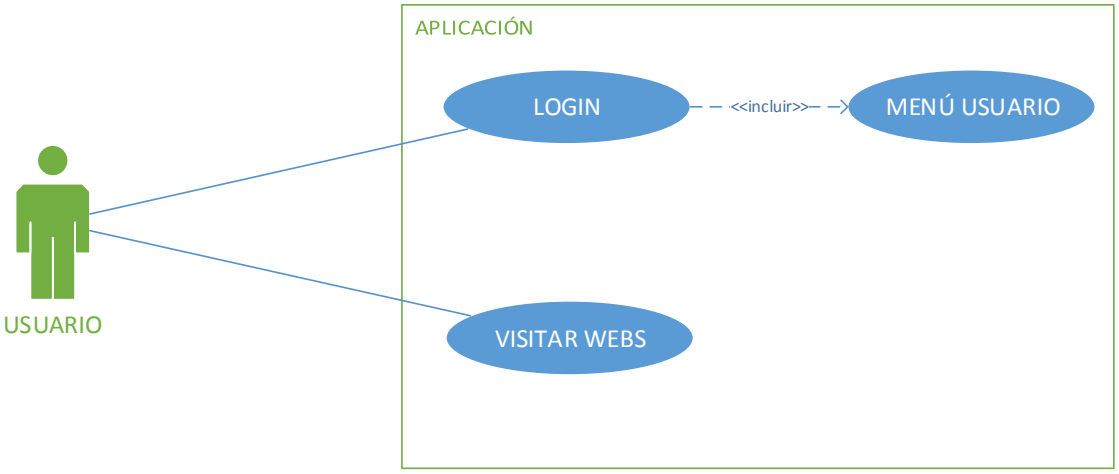


Figura 2.1 Casos de uso para el login.

Caso de uso para el login	
Descripción	Este primer diagrama, es el perteneciente al primer paso antes de acceder a la aplicación. Los usuarios ya estarán creados por el administrador, con lo cual, para poder acceder a ella, únicamente tendrán que insertar sus credenciales.
Precondición	Saber la existencia de esta web mediante un aviso a todos los usuarios de la editorial.
Opciones principales	Acceder a la web, para ver los menús y realizar las diferentes operaciones.
Opciones alternativas	Visitar las diferentes páginas oficiales de las plataformas.
Postcondición	El usuario accederá correctamente a la aplicación y podrá visualizar los diferentes menús disponibles para realizar operaciones.

2.2.1.2 Diagrama de Caso de Uso para el menú usuarios.

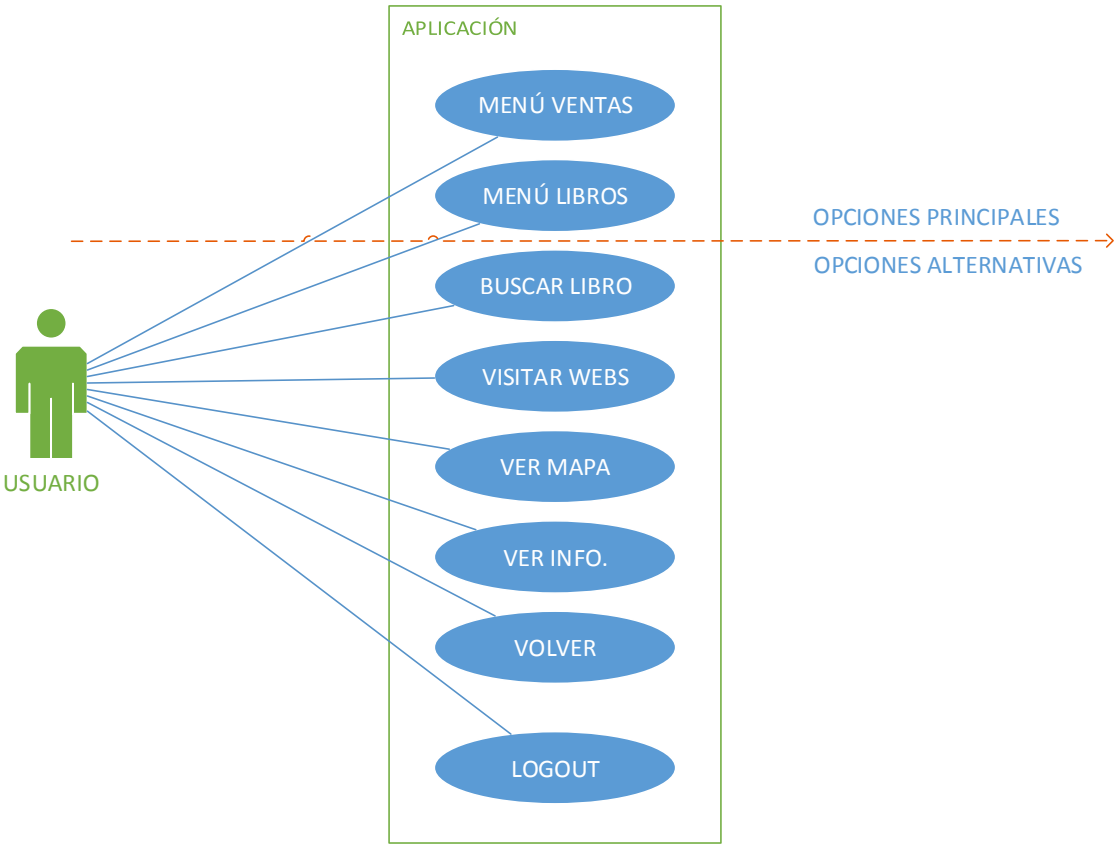


Figura 2.2 Casos de uso del menú de usuarios.

Caso de uso para el Menú principal	
Descripción	Una vez hayan accedido los usuarios, les aparecerá un menú en el cual podrán realizar diferentes acciones.
Precondición	Estar registrado en la base de datos interna. Insertar correctamente los datos de registro.
Opciones principales	○ Ir al menú de ventas ○ Ir al menú de sus libros
Opciones alternativas	○ Salir de la aplicación ○ Ver mapa de la localización de la empresa ○ Acceder a las diferentes páginas oficiales de las plataformas ○ Buscar un libro suyo
Postcondición	Según la opción que elija el usuario se accederá a un menú o a otro.

2.2.1.3 Diagrama de Caso de Uso para el menú ventas.

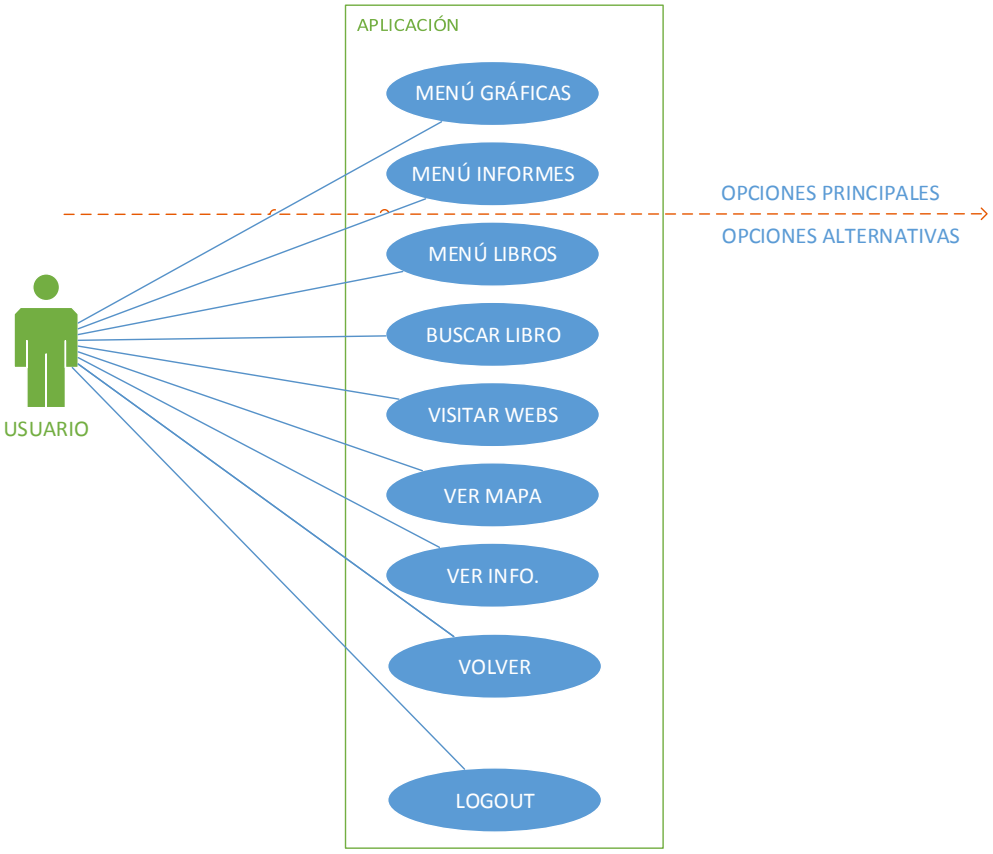


Figura 2.3 Casos de uso para el menú ventas

Caso de uso para el menú ventas	
Descripción	El usuario llegará a este menú a través del botón del menú ventas, donde dispondrá de diferentes funciones a realizar. Aquí se le mostrará la opción de visualizar sus ventas de la manera que desee.
Precondición	Haber pulsado el menú principal "Mis Ventas" Tener ventas de sus libros para poder visualizar.
Opciones principales	- Generar informe detallado de mis ventas - Generar gráfica para visualizar mis ventas
Opciones alternativas	○ Ir al menú libros ○ Salir de la aplicación ○ Ver mapa de la localización de la empresa ○ Acceder a las diferentes páginas oficiales de las plataformas ○ Buscar un libro suyo
Postcondición	- Si el usuario ha pinchado en generar informe detallado, este le llevará a una página donde podrá seleccionar cómo quiere generar el informe, en cuanto a fechas y demás. - Si el usuario ha pinchado en generar gráfica, le aparecerán diferentes opciones para realizar una gráfica y ver sus ventas.

2.2.1.4 Diagrama de Caso de Uso para el menú libros.

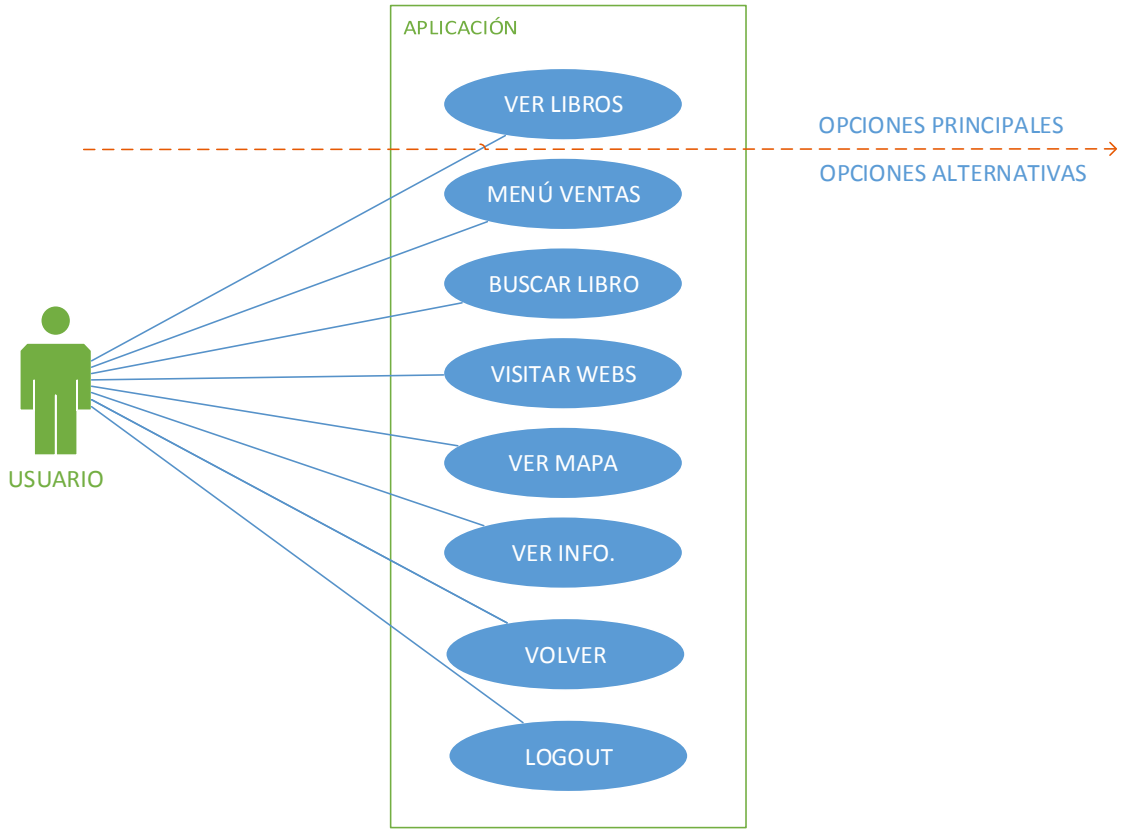


Figura 2.4 Casos de uso para el menú libros

Caso de uso para el menú libros	
Descripción	En este menú, el usuario podrá ver un listado con sus libros, donde se le dará la posibilidad de ver detalladamente la información de estos individualmente. Podrá realizar diferentes operaciones.
Precondición	Haber pulsado el menú principal de "Mis Libros"
Opciones principales	○ Ver información detallada de sus libros
Opciones alternativas	○ Ir al menú ventas ○ Salir de la aplicación ○ Ver mapa de la localización de la empresa ○ Acceder a las diferentes páginas oficiales de las plataformas ○ Buscar un libro suyo
Postcondición	El usuario obtendrá un listado con sus libros propios, los cuales podrá acceder a ver la información de estos. (En un futuro, serán los usuarios los que rellenen esta información)

2.2.1.5 Diagrama de Caso de Uso para crear informes.

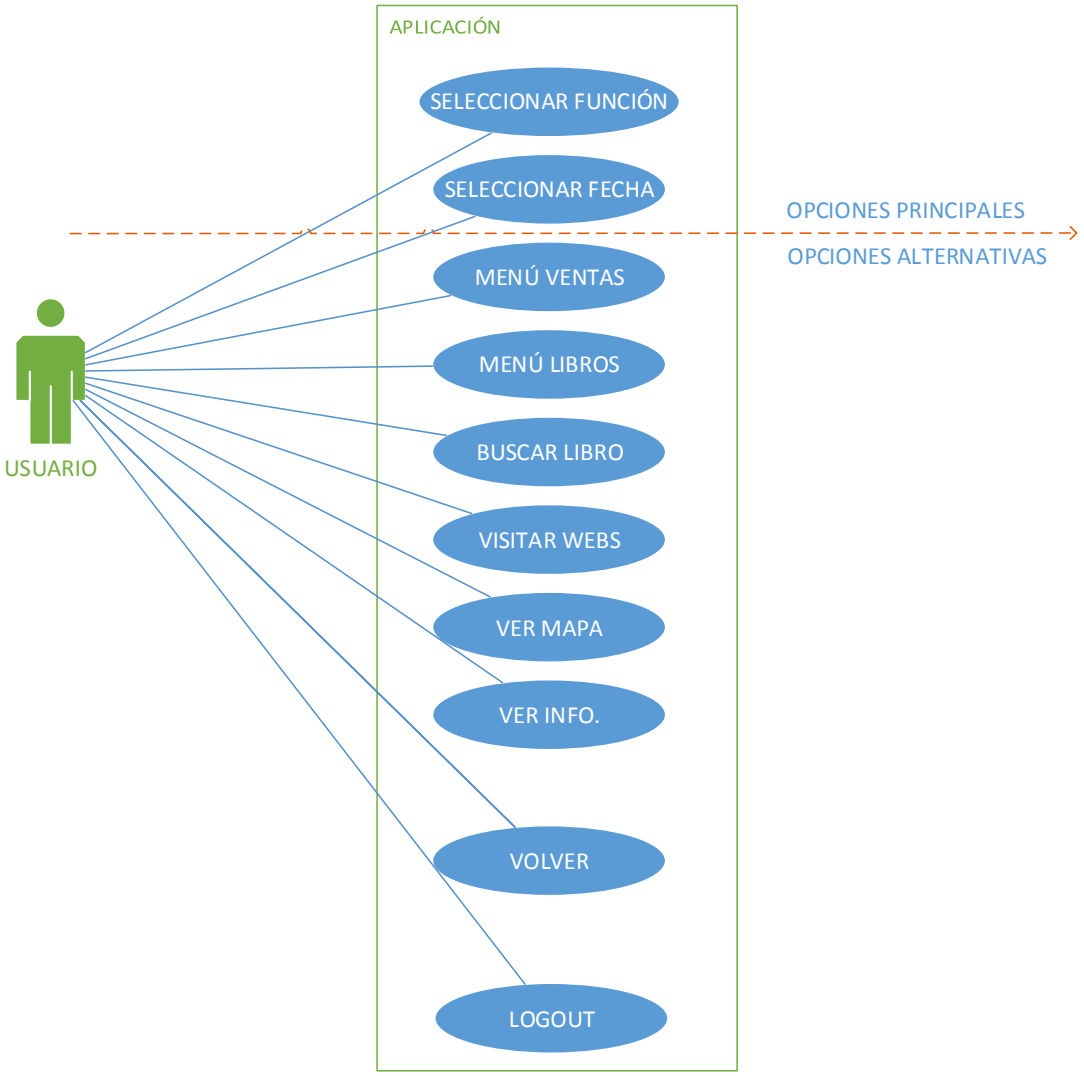


Figura 2.5 Casos de uso para crear informe

Caso de uso para crear informe	
Descripción	El usuario podrá seleccionar cómo quiere el informe en cuanto a rol, completo, del mes anterior, del año actual o seleccionar la fecha deseada.
Precondición	Seleccionar los filtros deseados. Tener ventas para poder realizar el informe.
Opciones principales	- Seleccionar rol para mostrar la gráfica. - Seleccionar fechas
Opciones alternativas	- Ir a otros menús - Salir de la aplicación - Acceder a las diferentes páginas oficiales de las plataformas
Postcondición	La aplicación le devolverá un documento .xlsx donde se recogerán las ventas de sus libros.

2.2.1.6 Diagrama de Caso de Uso para crear gráficas.

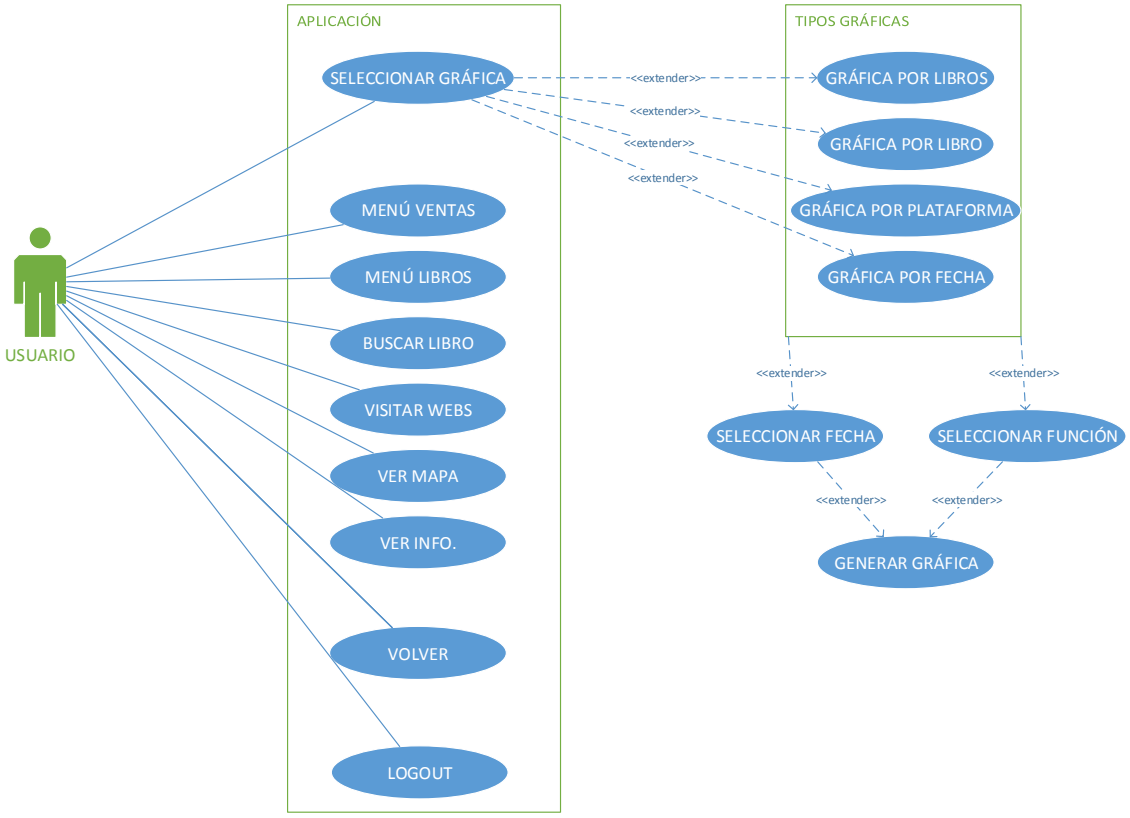


Figura 2.6 Casos de uso para crear gráfica

Caso de uso para crear gráfica	
Descripción	Aquí el usuario podrá visualizar sus ventas de un simple vistazo mediante gráficas. Será algo similar a lo anterior pero utilizando gráficas.
Precondición	Seleccionar los filtros deseados. Tener ventas para poder realizar el informe.
Opciones principales	Seleccionar el tipo de gráfica deseada. Seleccionar fechas. Seleccionar rol del usuario para ver los libros que desee.
Opciones alternativas	- Ir a otros menús - Salir de la aplicación - Acceder a las diferentes páginas oficiales de las plataformas
Postcondición	La aplicación le devolverá al usuario una gráfica por pantalla donde le mostrará el resultado de la consulta.

2.1.2 Casos de uso para el Backend.

Aquí se mostrarán los casos de uso para la parte del Backend, donde la parte más importante será la inserción de las ventas en la base de datos a través de ficheros .csv y .txt y la administración de la base de datos.

En la administración de los datos, aparecerá la opción de dar de alta, modificar, lista o eliminar un registro. Dicho esto, para mostrar los diagramas de casos de uso para la parte de administración, con mostrar un diagrama será suficiente, ya que todos los demás serán similares.

2.2.2.1 Diagrama de Caso de Uso para el Login.

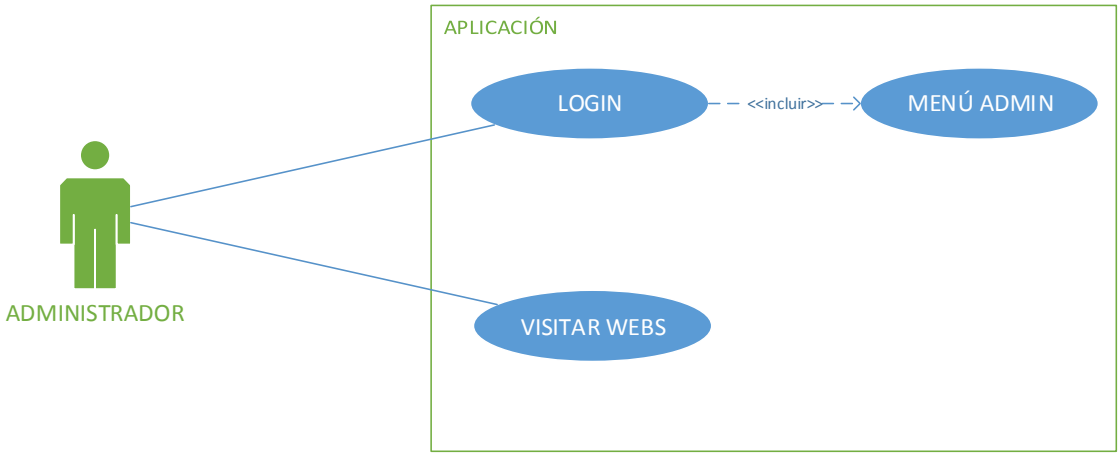


Figura 2.7 Casos de uso para el login.

Caso de uso para el login	
Descripción	Los administradores ya estarán creados anteriormente, con lo cual para poder acceder a la aplicación, únicamente tendrán que insertar sus credenciales correctamente.
Precondición	Saberse los datos de un usuario administrador para poder entrar.
Opciones principales	Acceder a la web, para ver los menús y realizar las diferentes operaciones.
Opciones alternativas	Visitar las diferentes páginas oficiales de las plataformas.
Postcondición	El administrador accederá correctamente a la aplicación y podrá visualizar los diferentes menús disponibles para realizar operaciones

2.2.2.2 Diagrama de Caso de Uso para el menú de los administradores.

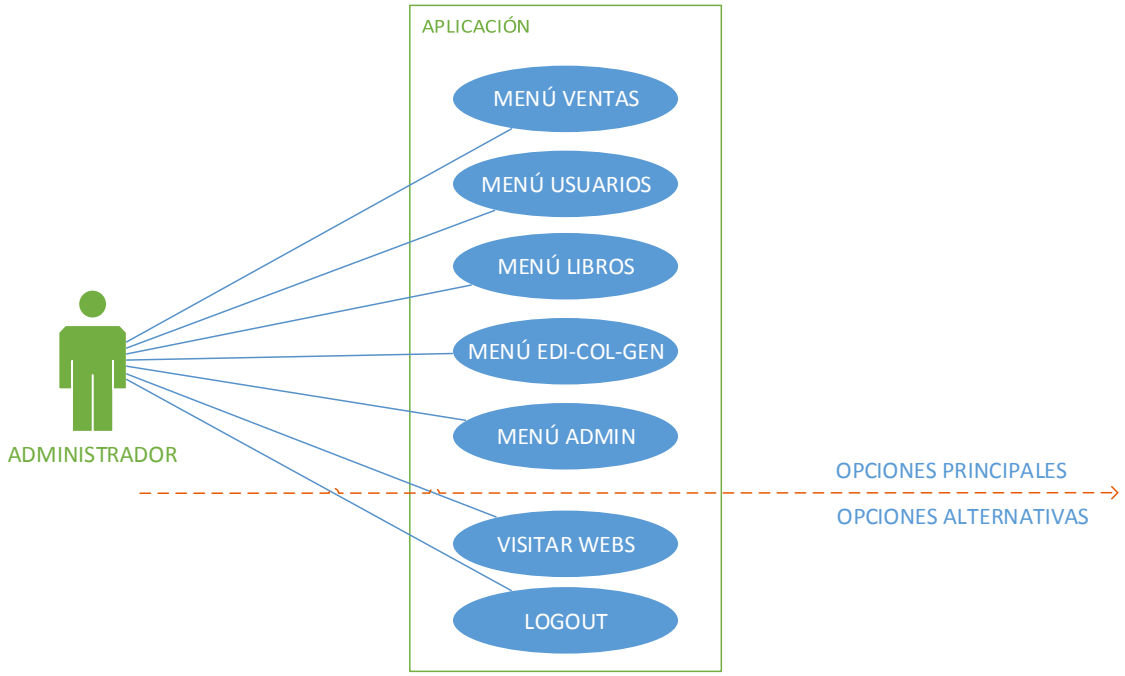


Figura 2.8 Casos de uso del menú de los administradores.

Caso de uso menú administrador	
Descripción	Este diagrama muestra las diferentes opciones que tendrá el administrador una vez haya entrado a la aplicación.
Precondición	Estar registrado como administrador en la base de datos interna. Haber ingresado correctamente los datos del usuario administrador.
Opciones principales	Elección de acción a realizar: - Administrar una venta - Administrar usuarios - Administrar libros - Administrar editoriales/colecciones/géneros - Administrar usuarios administradores
Opciones alternativas	- Visitar las diferentes páginas oficiales de las plataformas. - Salir de la aplicación
Postcondición	Según el menú seleccionado, se accederá a administrar unos datos u otros, los que el administrador necesite en ese momento.

2.2.2.3 Diagrama de Caso de Uso para el menú de las ventas.

Para este caso, voy a realizar los diferentes niveles para mostrar los casos de uso para el menú ventas ya que va a quedar más claro.

En el nivel cero, una vez que el administrador ha seleccionado el menú ventas, aparecerán 3 botones donde este podrá seleccionar lo deseado.

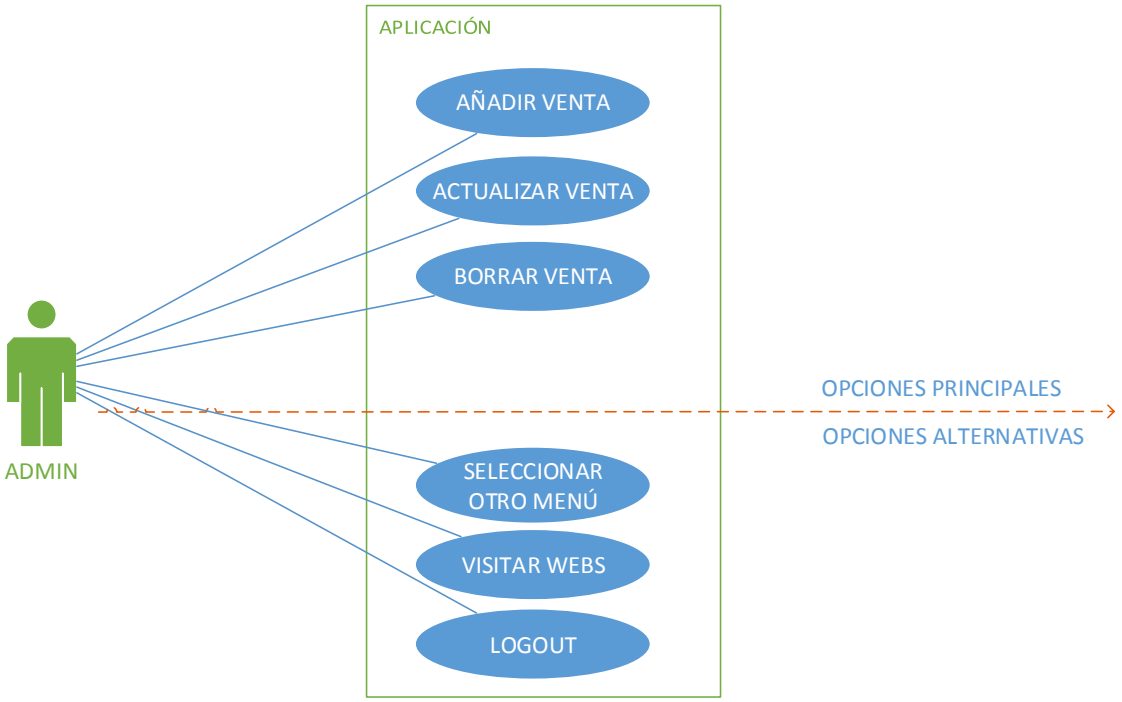


Figura 2.9 Casos de uso del menú de las ventas.

Caso de uso menú ventas del administrador	
Descripción	En el caso de seleccionar el menú ventas, al administrador le aparecerán 3 opciones: - Añadir, - Actualizar, -Eliminar. Este seleccionará lo deseado.
Precondición	Estar registrado como administrador en la base de datos interna. Haber ingresado correctamente los datos del usuario administrador. Haber seleccionado el menú ventas
Opciones principales	Elección de acción a realizar: - Añadir una venta. - Actualizar una venta. - Eliminar una venta.
Opciones alternativas	- Visitar las diferentes páginas oficiales de las plataformas. - Salir de la aplicación. - Seleccionar otro menú.
Postcondición	Según el menú seleccionado, se accederá a realizar una operación u otra.

En el nivel uno, una vez que el administrador ha seleccionado una de las tres opciones, aparecerá lo correspondiente a cada acción:

- Añadir venta: El administrador seleccionará los archivos deseados para subirlos a la base de datos para reflejar las ventas.
- Actualizar venta: en el caso de que una venta ya subida, tenga que ser actualizada en cuanto a 'x' campos, se realizará desde aquí. El administrador deberá indicar el periodo de fechas de las ventas a actualizar así como la plataforma de las mismas.
- Eliminar venta: en el caso de que después de subir una venta, la plataforma de la misma nos indique que los datos de esa venta son incorrectos, se procederá a eliminar todos los registros de esta.

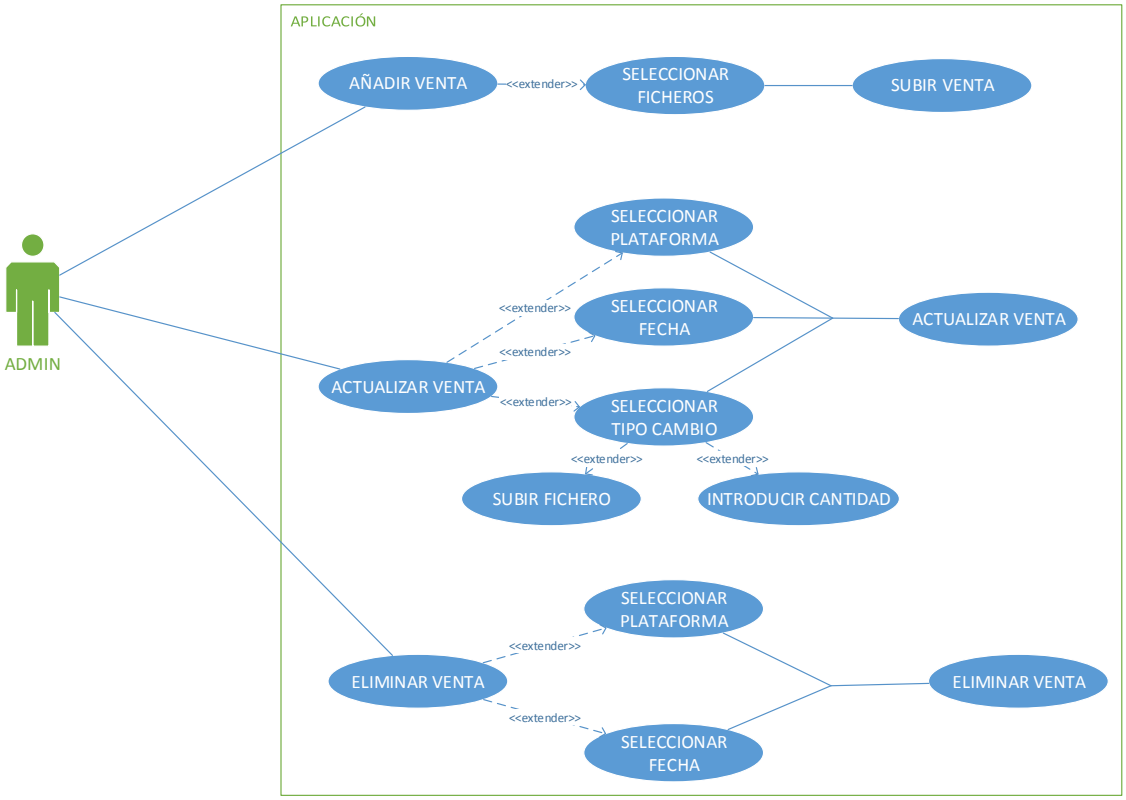


Figura 2.10 Casos de uso de funciones ventas.

Caso de uso menú ventas del administrador	
Descripción	Se procederá a administrar las ventas de los usuarios.
Precondición	Subir venta: Haber seleccionado, los archivos a subir. Actualizar venta: Seleccionar periodos y plataforma. Eliminar venta: Seleccionar la venta a eliminar.
Opciones principales	- Añadir una venta - Actualizar una venta - Eliminar una venta
Opciones alternativas	- Visitar las diferentes páginas oficiales de las plataformas. - Salir de la aplicación - Seleccionar otro menú
Postcondición	Según el menú seleccionado, se accederá a realizar una operación u otra.

2.2.2.3 Diagrama de Caso de Uso para la administración de los datos.

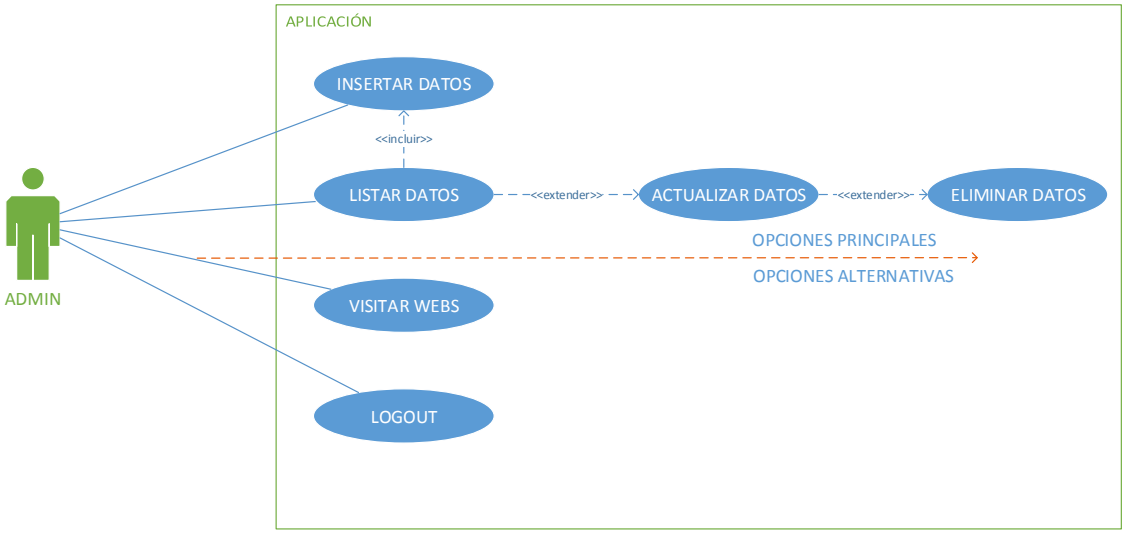


Figura 2.11 Casos de uso para la administración.

Caso de uso menú administrador	
Descripción	Este diagrama muestra las diferentes opciones que tendrá el administrador a la hora de tratar los diferentes datos de la aplicación.
Precondición	Seleccionar la entidad de la cual se quiere administrar datos
Opciones principales	Elección de acción a realizar: - Alta de datos. - Baja de datos. - Listar datos. - Modificar datos.
Opciones alternativas	- Visitar las diferentes páginas oficiales de las plataformas. - Salir de la aplicación. - Seleccionar el menú deseado.
Postcondición	Según el menú seleccionado, se accederá a administrar unos datos u otros, los que el administrador necesite en ese momento.

3.- DISEÑO

En este capítulo se explicará la metodología utilizada, la estructura de la aplicación y el modelo relacional.

3.1 Metodología

Dentro de la Ingeniería del software existen varios modelos para llegar a la construcción final de un producto de software y optimizar el desarrollo del mismo. Cada modelo tiene ventajas y desventajas de tal forma que para realizar un buen proyecto, cada usuario tiene que elegir el que mejor se adapte a sus necesidades, tanto para la creación, elaboración y mantenimiento.

Una metodología es un conjunto de técnicas las cuales te permiten realizar correctamente cada una de las fases del ciclo de vida de tu proyecto. Al hacer uso de estas técnicas tienes que detallar y concretar varios aspectos antes de comenzar.

En mi caso, al realizar una aplicación web, la metodología me determinará qué procesos he de seguir para gestionar y administrar el proyecto correctamente, de esta forma llegaré al objetivo y conseguiré los requisitos del producto.

Después de hacer un estudio previo de cómo iba a ser mi aplicación, y qué pasos iba a seguir para implementarla, opté por utilizar la metodología basada en prototipos.

Decidí utilizar esta metodología porque es un método iterativo para desarrollar software eficazmente. Además, inicialmente no tenía claros los requisitos de toda la aplicación, como la interfaz de esta, el cómo insertar los datos de los informes, cómo iban a ser los informes generados, etc. Esto me hizo pensar en realizar un prototipo de cómo va a ser la aplicación y después de tener una idea, ir modificándola.

Esto conllevaba que el cliente (el responsable de la empresa), realizara un alto grado de participación, ya que a menudo ha tenido que aportar opiniones y requisitos.

Por otro lado, los prototipos son el mejor medio de comunicación en cuanto a dar una pre-visualización al cliente, ya que realizando un prototipo, el cliente va a ver el efecto al instante de cómo se va creando la aplicación, de tal forma que este te puede ir aportando ideas.

En mi caso por ejemplo, a lo largo de la carrera no he realizado ninguna asignatura que sea de diseño, con lo cual mis ideas son escasas, el cliente (en mi caso el encargado de este tema en la empresa), es el que a menudo trabaja con este tipo de aplicaciones y el cual conoce las necesidades que pueden tener los usuarios.

Este es un proceso lento, ya que es un proceso adelante-atrás por los ajustes que se van realizando constantemente.

Al utilizar este método, las etapas del ciclo de vida pueden ser las siguientes:

- Análisis de los requisitos generales del sistema.
- Diseño, desarrollo e implementación del prototipo.
- Prueba del prototipo.
- Refinamiento del prototipo.
- Refinamiento de las especificaciones del prototipo.
- Diseño e implementación del sistema final.
- Mantenimiento.

Cabe destacar que este modelo es fácilmente modificable y ampliable en cualquier momento.

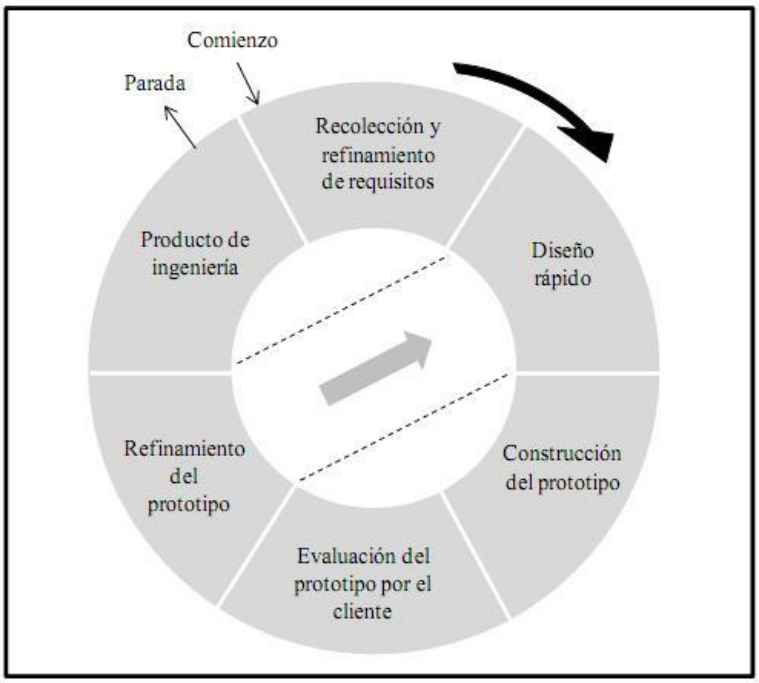


Figura 3.1 Elaboración de un prototipo.

Este modelo tiene ventajas pero también tiene desventajas a la hora de trabajar con él. Por un lado, es bueno porque reduce la incertidumbre y el riesgo, ya que el cliente va viendo lo que se va realizando, esto conlleva una mejor comunicación entre desarrollador y cliente. Es útil cuando el cliente (como en mi caso) conoce los objetivos generales para el software, pero no identifica los requisitos detallados de entrada o salida.

Por otro lado, este modelo posee una serie de desventajas, al realizar un prototipo de cómo puede ser la aplicación, el cliente ve una primera versión la cual ha sido construida sin detalles, con lo que puede llevar a la desilusión de este ya que no ve lo que quiere.

Para concluir, es un modelo fácil de utilizar y de modificar en el momento que se desee, analizando las alternativas y en el caso de que no se cubran las expectativas del cliente, se procede a repetir las etapas para que el sistema desarrollado sea de calidad.

Cada semana, iba realizando diferentes apartados de la aplicación. De esta forma le iba mostrando al cliente los resultados y realizaba pruebas para ver si el funcionamiento era el correcto.

Primero realicé un planteamiento de la base de datos, seguido la creación de esta. Los pasos siguientes fueron la elaboración del backend, con la administración de la bbdd y seguido la subida de ficheros de las ventas. Por último, la creación del frontend, empezando con la generación de informes y seguido la creación de las gráficas junto con el diseño de las interfaces.

Las fases que he seguido para la implementación de cada paso de la aplicación son las marcadas por la metodología de prototipos, para cada apartado que engloba la totalidad de una parte de la aplicación realizaba todas las fases.

- Análisis de los requisitos generales del sistema: Tras sucesivas reuniones con los responsables de este tema, reuní los requisitos que ellos creían convenientes (los expuestos arriba) y si no iba aportando yo ideas. Todas las partes realizadas, antes consultaba con los compañeros para ver cómo sería mejor implementarlo.

- Diseño, desarrollo e implementación del prototipo: en esta fase desarrollaba una primera interfaz de cómo podría ser esa parte de la aplicación, en cuanto a estructura del menú y demás, si esta idea les parecía bien le añadía el css.

- Prueba del prototipo: aquí realizaba sucesivas pruebas para ver que el funcionamiento era el adecuado y no hacia cosas raras, acto seguido le decía a mi compañero que utilizase la aplicación para realizar determinadas tareas.

- Refinamiento del prototipo: en el caso de mostrarle a mis compañeros y estos dar un visto nulo o una posible mejora, ellos me daban su opinión de cómo situar los botones o las opciones y llegábamos a una decisión.

- Refinamiento de las especificaciones del prototipo: si lo que habíamos pensado en un primer momento, después de realizar la implementación no nos convencía, cambiábamos las especificaciones del prototipo y se ajustaba a lo deseado.

- Diseño e implementación del sistema final: una vez que estaban hechas todas las partes, lo que hice fue adaptar todos los css para que fuesen acordes unos con otros, y así al cambiar menús no se notase, ya que haría un efecto poco deseable.

Todas estas fases las hemos ido realizando conforme la aplicación iba cogiendo forma, ya que como he mencionado anteriormente, en la carrera no he dado una asignatura de diseño, y son ellos los que más han tratado con este tipo de aplicaciones.

3.2 Estructura del sitio web

Inicialmente mi intención era seguir el modelo de la página de Leer-e de la empresa, para que en el caso de insertar la funcionalidad en un futuro en ella, siguiese los mismos patrones. Me puse a hacer los bocetos y después de hacerlos en un PowerPoint y enseñárselos (ver Figura 3.2), me comunicaron que no tenía por qué ir en la propia web, y que hiciese otro boceto.

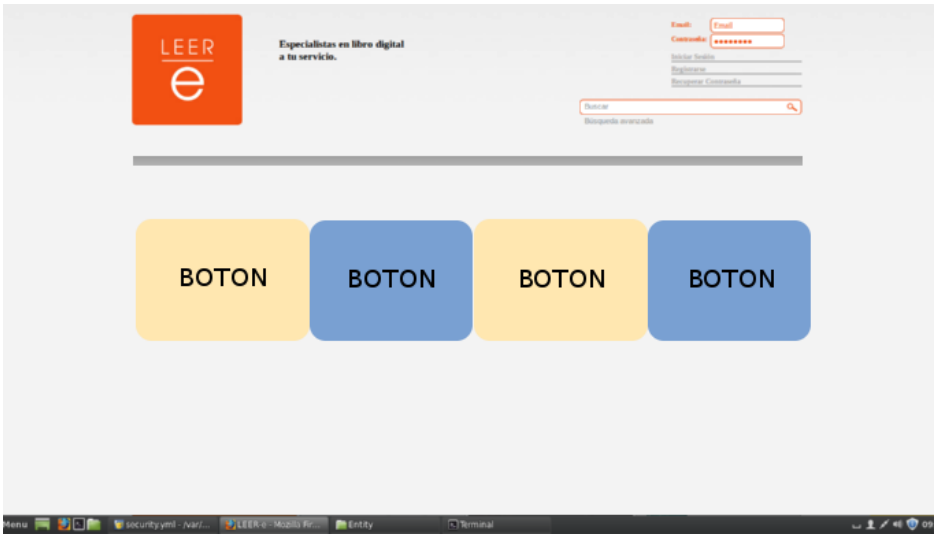


Figura 3.2 Diseño del primer prototipo.

Esto es debido a que esta aplicación la controlan desde otra empresa y no se tiene acceso a cambiar lo que desean, con lo cual, me decanté por un menú sencillo en el que ibas pulsando los diferentes botones y te ibas moviendo por los sub-menús.

En un primer momento lo veíamos bien pero nos parecía un poco 'simple', ya que la idea es que sea un sitio público y que los clientes queden satisfechos, con lo cual busqué diferentes modelos de menús en una web en la que ofrece una gran cantidad de menús gratuitos. Finalmente con el responsable de allí en este tema, nos decidimos por el menú actual para la parte del frontend, ya que se acercaba a lo que íbamos buscando.

Este proceso no es sencillo porque una vez seleccionado, tienes que adaptarla a tus necesidades y no siempre es fácil.

En cuanto al tema de la interfaz, en mi opinión, visualmente es un sitio web poco complejo, quitando el menú que tiene un poco de interactividad, pero la funcionalidad queda clara, ya que es el objetivo que busca la empresa debido al tipo de usuarios que la van a utilizar

Además, en cualquier momento esta puede ser reemplazada por otra interfaz sin ningún tipo de problema, ya que al utilizar las plantillas de Twig, solo habría que cambiar la plantilla base, y de esta forma obtendríamos la interfaz deseada.

La Figura 3.3 muestra un esquema de cómo es la interfaz de la aplicación para los usuarios, la parte de administración es muy parecida.

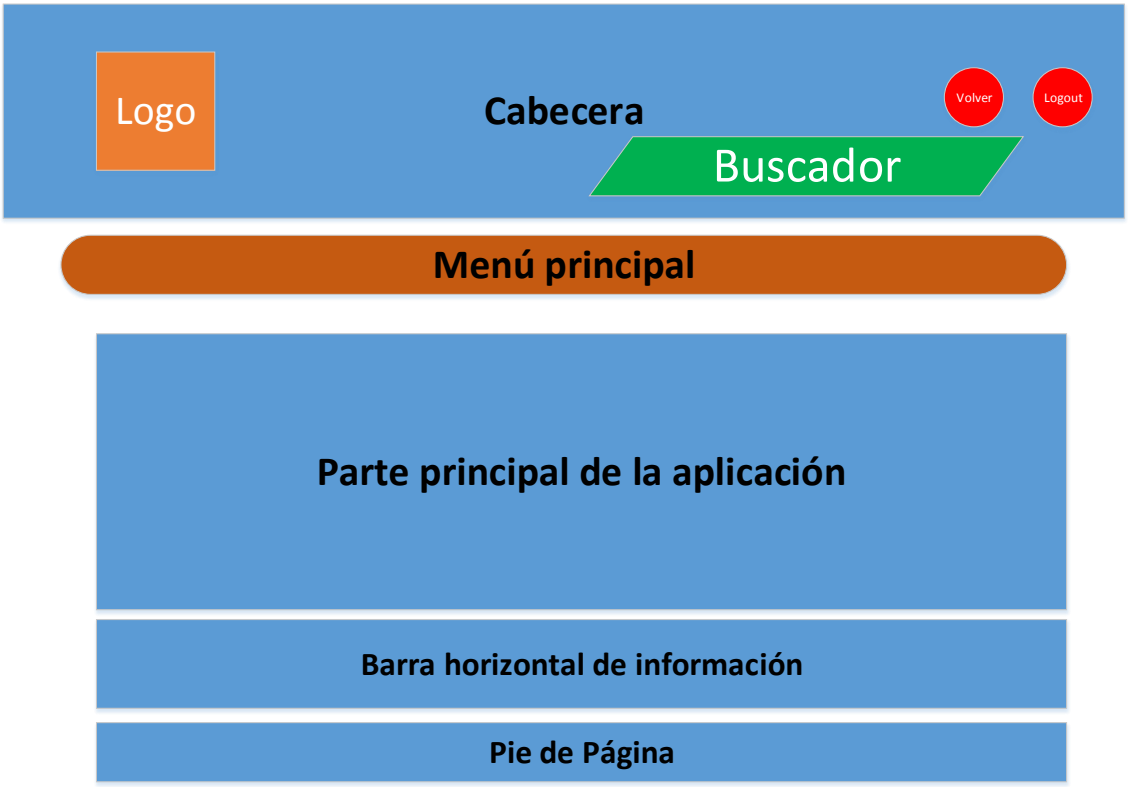


Figura 3.3 Esquema del prototipo de la interfaz del frontend.

Voy a describir la interfaz del frontend desde la parte superior hasta la parte inferior, dividiéndose en diferentes cajas. (Imagen anterior)

- **Cabecera**: situada en la parte superior, está constituida por el logo, el cual lo he realizado con GIMP2, la opción de salir de la aplicación (el botón logout), un botón para volver al menú anterior y un buscador que facilitará a los usuarios la búsqueda en la base de datos de los libros que deseen (los suyos propios).
- **Menú principal**: situado justo debajo de la cabecera, es un menú en el cual aparecen las distintas opciones para administrar y alimentar nuestra base de datos, en el caso del backend, y consultar ventas en el caso del forntend:
-

Menú principal backend	*Menú principal frontend*
Ventas	**Mis ventas**
Libros	**Mis libros**
Usuarios	
Editoriales, Colecciones y Géneros	
Usuarios Administradores	

- **Parte principal de la aplicación:** es la parte central donde se irán mostrando las diferentes páginas al ir navegando por la web dependiendo de los botones que se vayan presionando.
- **Barra horizontal de información:** es una barra horizontal que para el caso de los usuarios, muestra la opción de mostrar un mapa con la dirección de la empresa y otra con los datos de esta.
- **Pie de página:** Es la parte inferior de la página donde por un lado se muestran los distintos logos de cada plataforma de tal forma que se pueda acceder a sus páginas directamente, y ofrece un botón para ir al principio de la página.

3.3 Diseño de los casos de uso.

En este apartado voy a analizar los diagramas de casos de uso más significativos mediante diagramas de secuencia, en los cuales queda reflejado claramente el proceso para realizar una acción en la web. Estos diagramas ayudan mucho a la hora de realizar la implementación, porque te van mostrando los diferentes pasos que tienes que ir implementando.

Por un lado mostraré los diagramas referentes al Backend:

Diagrama de secuencia del Login:

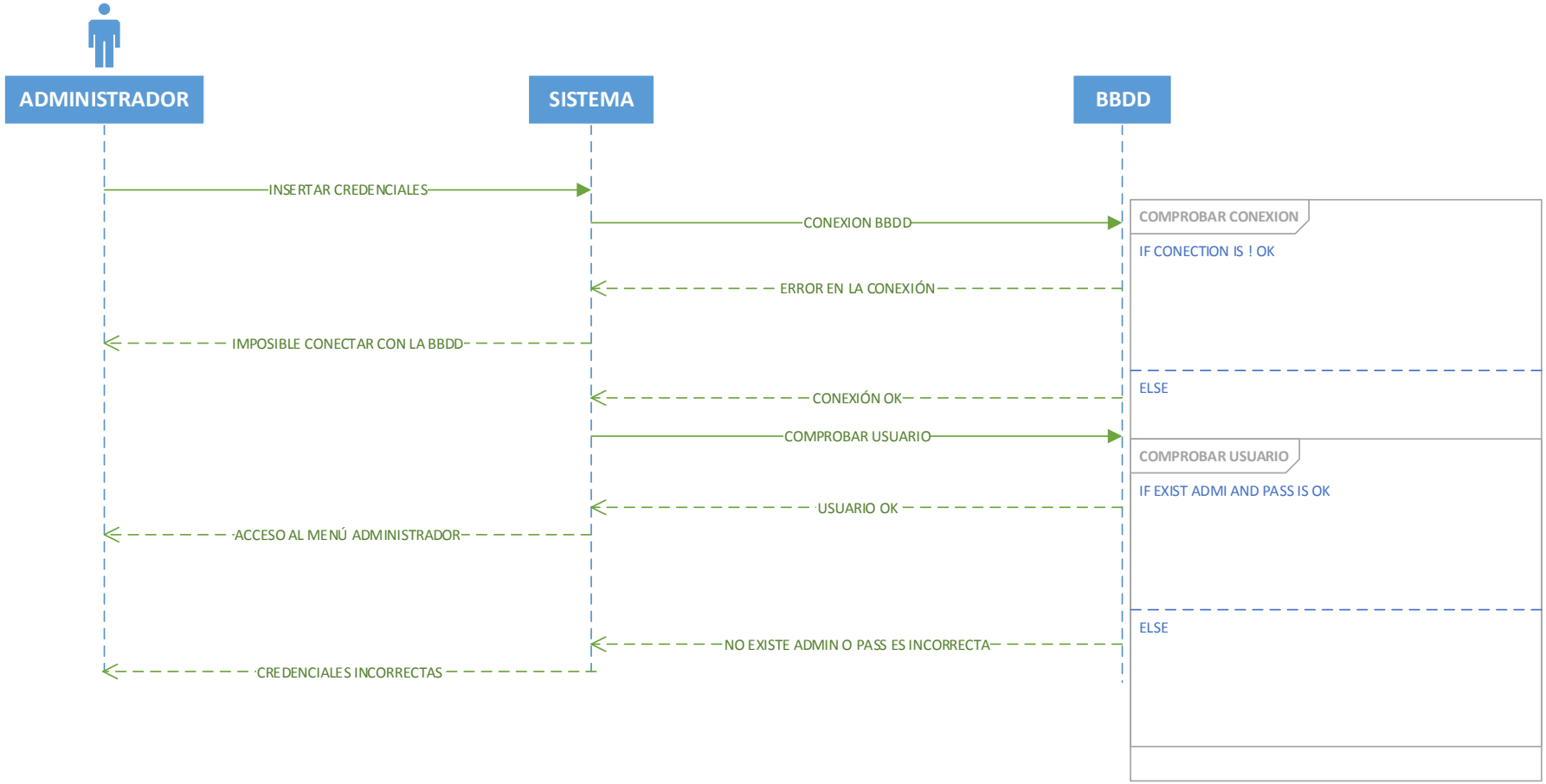


Figura 3.4 Diagrama secuencia del Login

Diagrama de secuencia de insertar, actualizar o eliminar datos:

- Insertar:

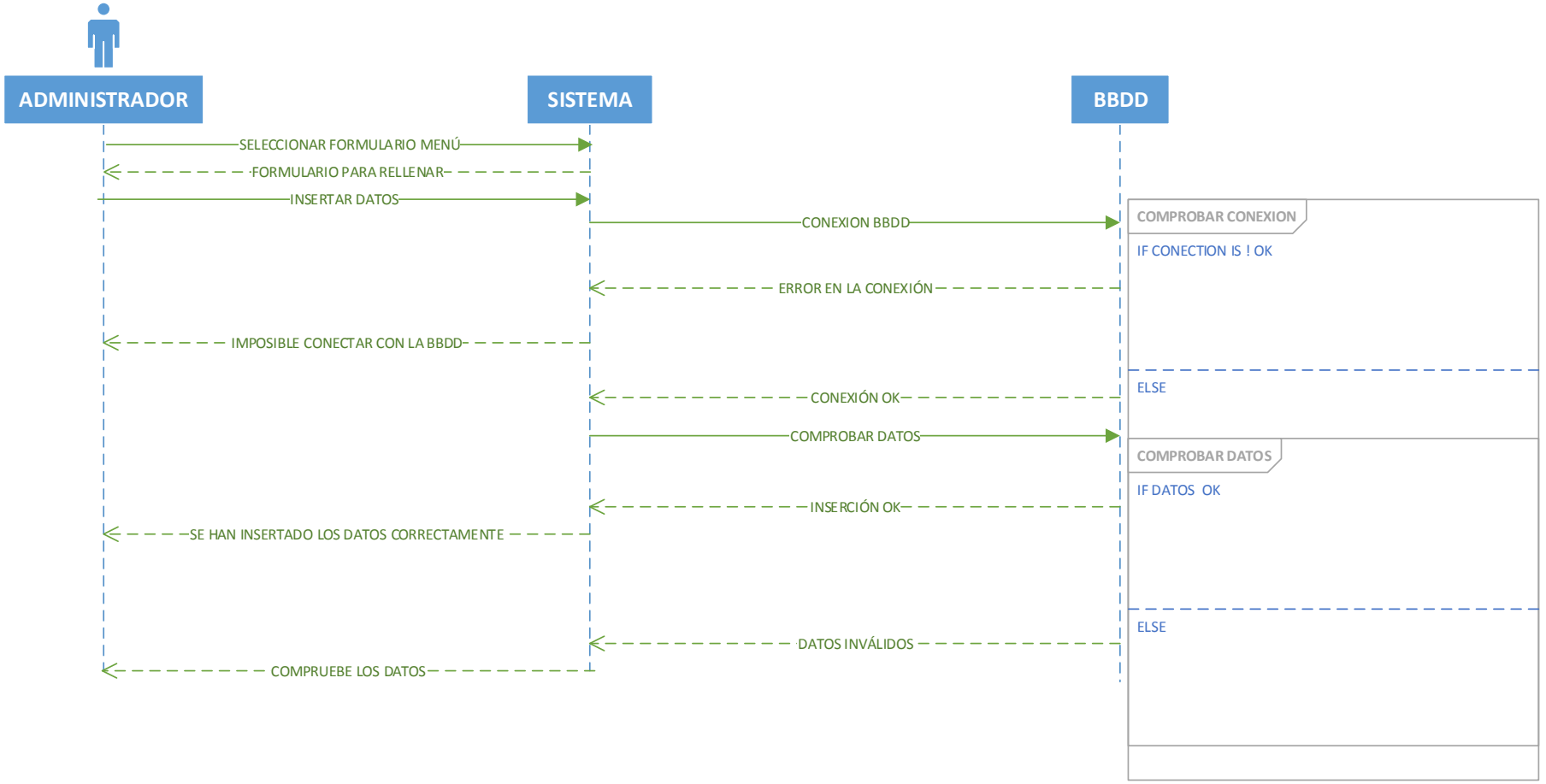


Figura 3.5 Diagrama secuencia de Insertar

- Modificar:

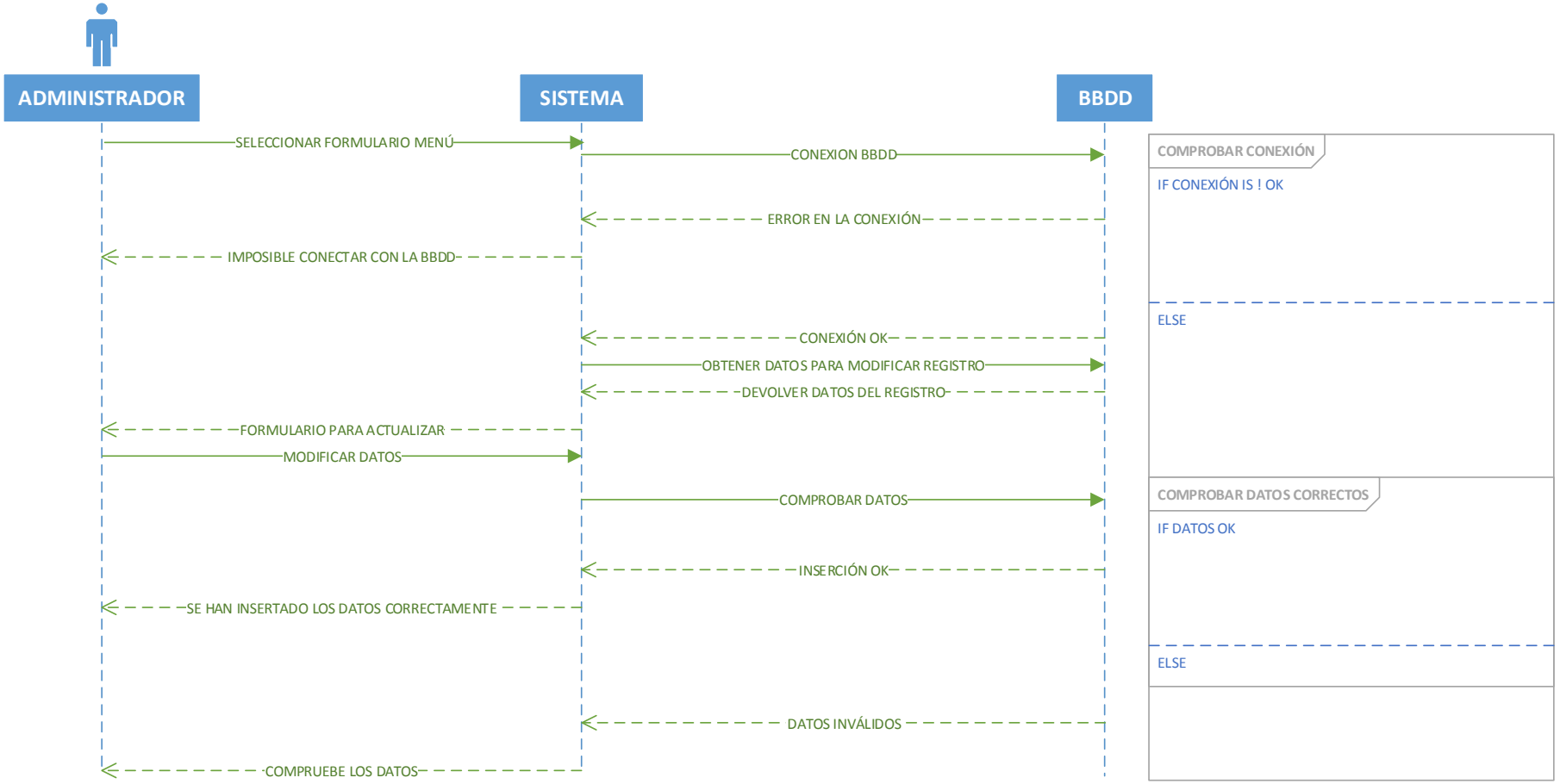


Figura 3.6 Diagrama secuencia de Modificar

- Eliminar:

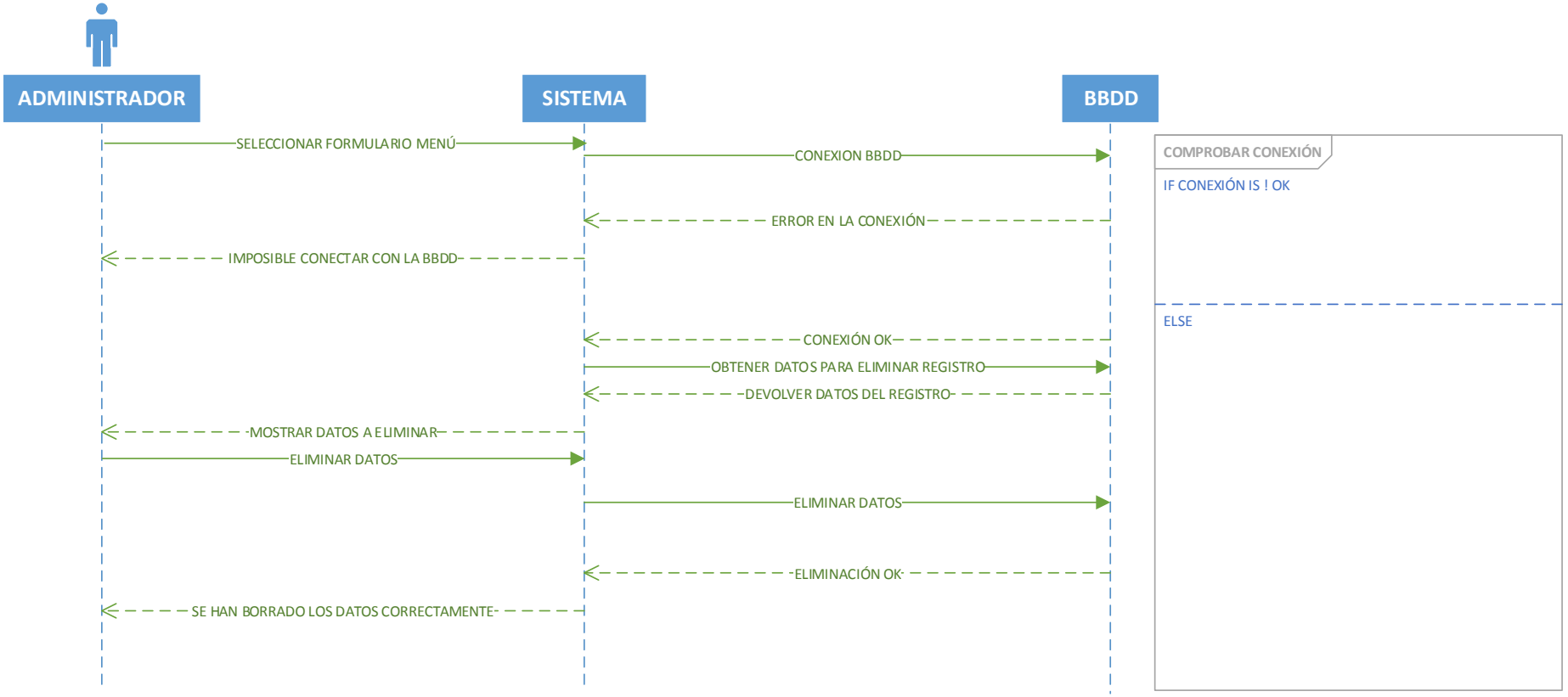


Figura 3.7 Diagrama secuencia de Eliminar

Diagrama de secuencia de subida de ventas:

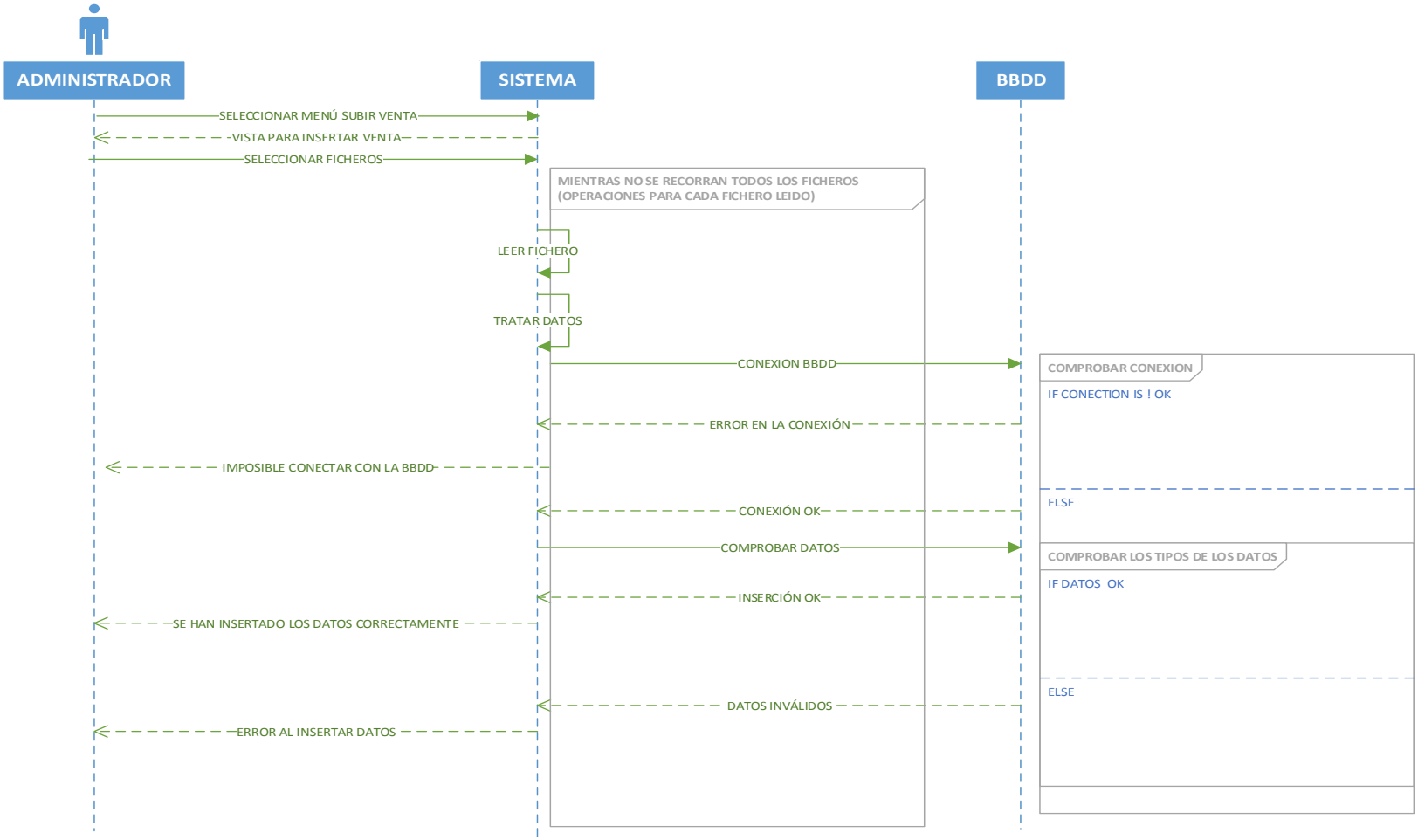


Figura 3.8 Diagrama secuencia subida de ventas

Por otro lado mostraré los diagramas referentes al Frontend:

Diagrama de secuencia de descarga de informes:

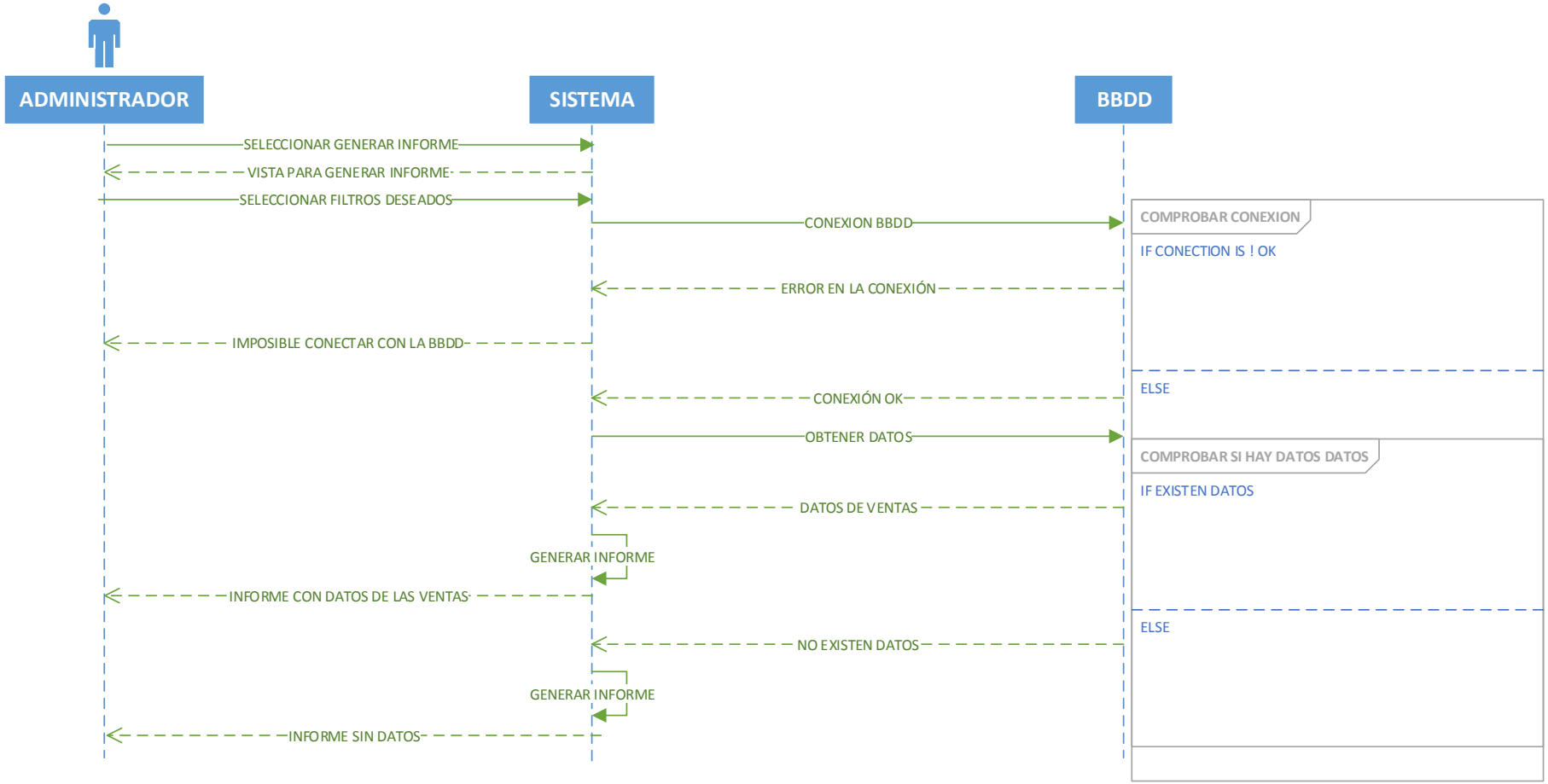


Figura 3.9 Diagrama secuencia descarga de informes

Diagrama de secuencia de vista de ventas mediante gráficas:

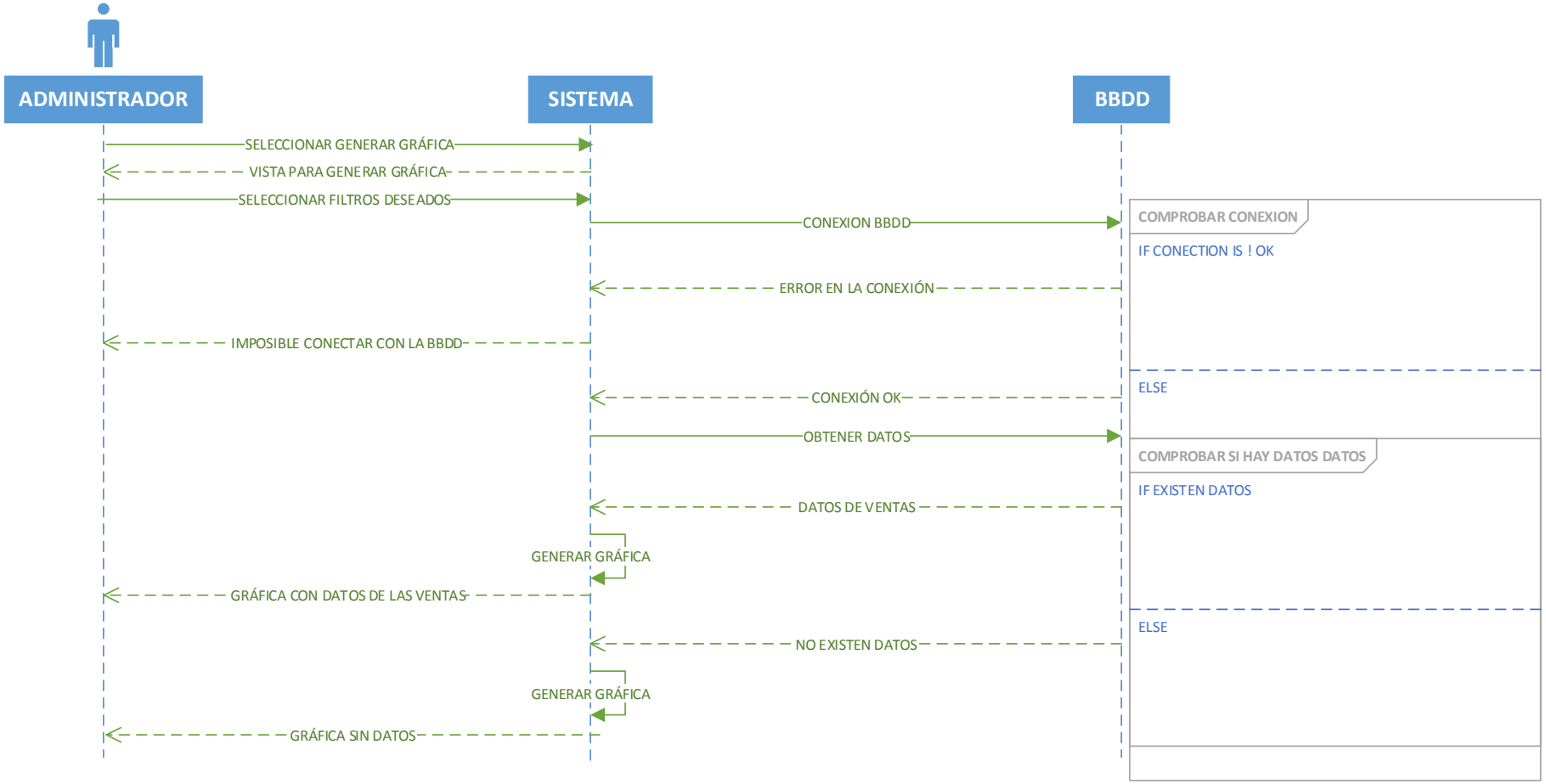


Figura 3.10 Diagrama secuencia ver gráficas

Diagrama de secuencia del buscador de libros:

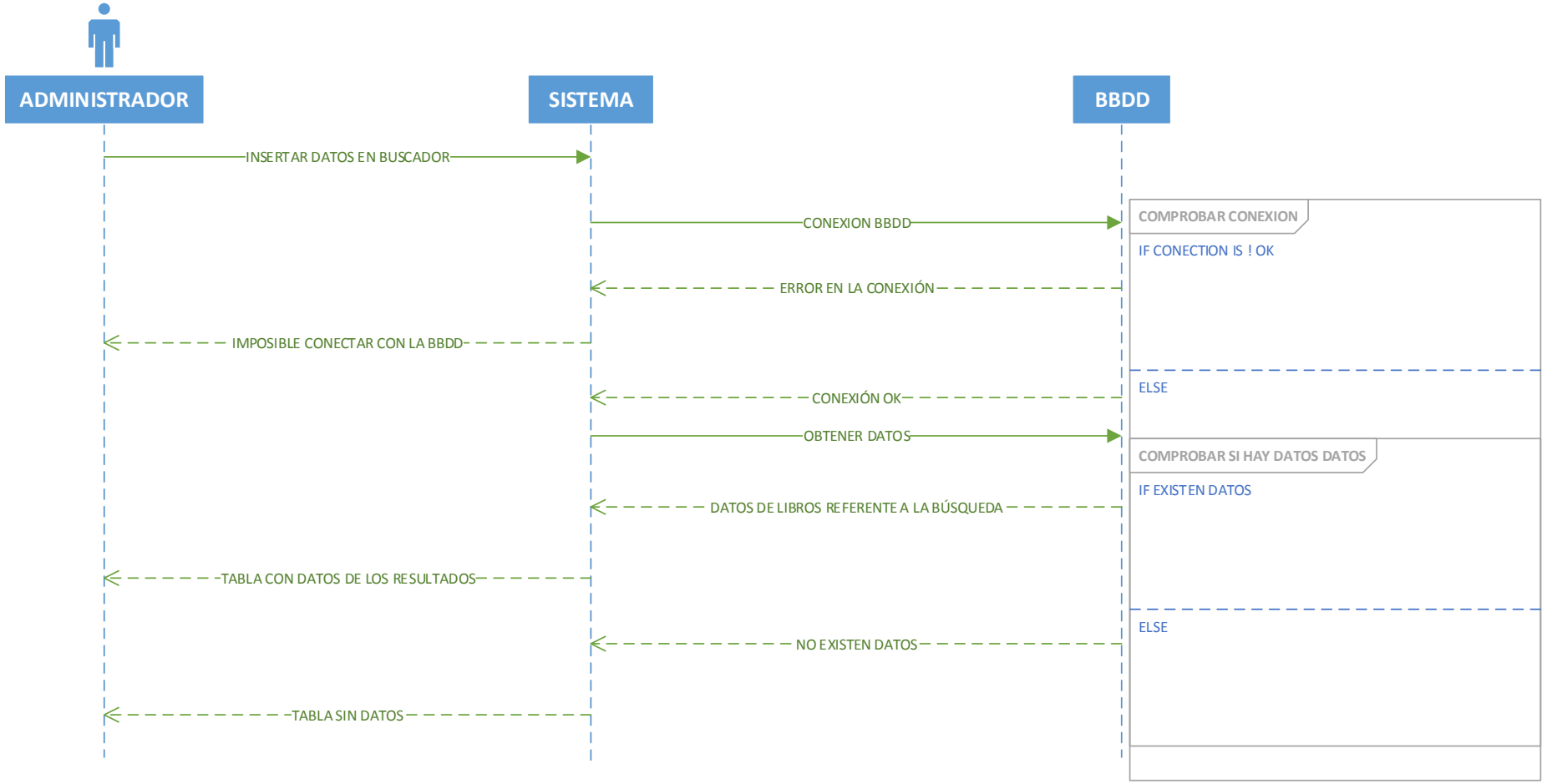


Figura 3.11 Diagrama secuencia buscar Libros

3.4 Modelo Relacional (Base de datos)

Como ya he comentado anteriormente, este proceso es de los más importantes y el que más detenidamente hay que pensar, ya que un mal planteamiento te puede acarrear sucesivos problemas y posteriormente tengas que realizar numerosas modificaciones. Lo realicé junto a mis compañeros de la empresa los cuales tenían los conocimientos de la información.

Después de adquirir un buen manejo con Symfony, nos pusimos manos a la obra para pensar en cómo crear el modelo relacional.

Primero, mis compañeros y yo, pensamos en la parte de los usuarios, inicialmente diferenciamos entre usuario Agente y usuario Autor, ya que los agentes no tenían libros propios sino libros de un determinado autor. Aunque los datos que queríamos almacenar eran similares unos de otros, creímos que lo mejor era diferenciar entre Agentes y Autores, y así lo hicimos.

Después, seguimos con los datos de los libros. Para concretar esta información, observamos los ficheros Excel que poseen ellos con los datos de todos sus libros, como anteriormente trabajaban con libros en papel, muchos datos eran innecesarios, como, peso, tipo de papel, etc. Por ello nos decantamos por los datos principales que puede tener un libro en formato electrónico.

Y por último, ojeamos los diferentes documentos que obteníamos de las diferentes plataformas sobre la información de las ventas. En algunos casos, estos informes, traen numerosos campos que no se utilizan, con lo cual nos quedamos con los datos que van a dar información tanto al cliente como al administrador.

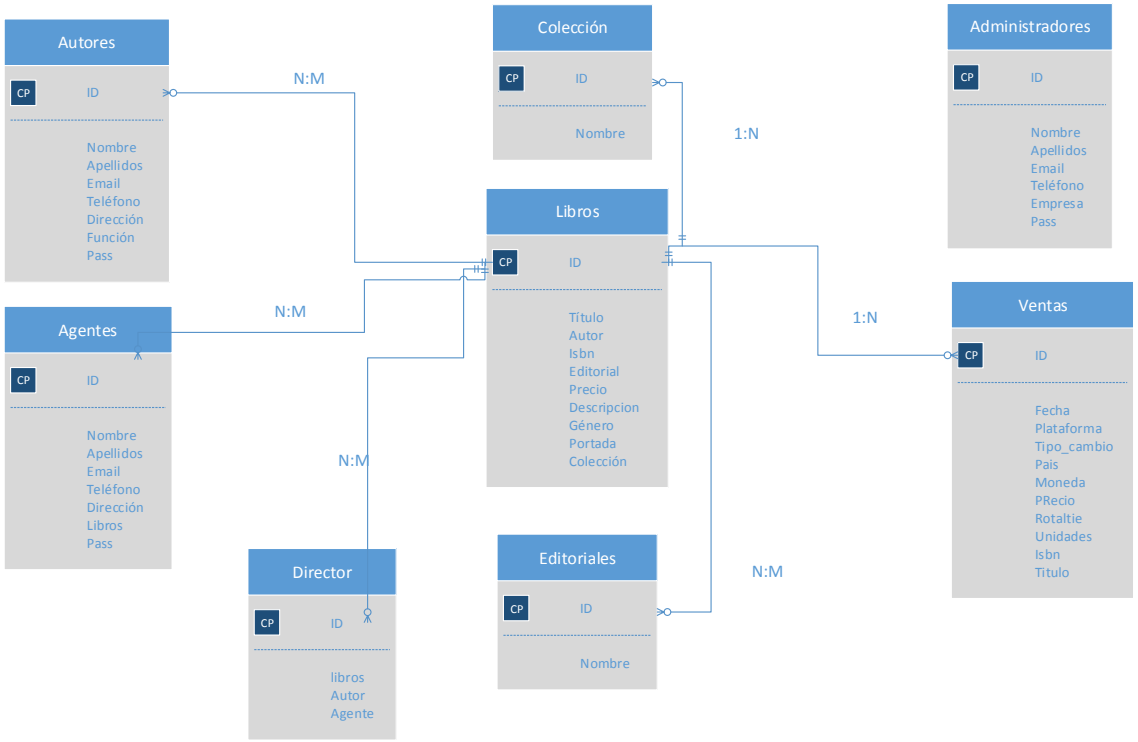


Figura 3.12 Primer diseño de la base de datos.

Después de tener todo pensado, realice con Doctrine la creación de la base de datos. Symfony trae una herramienta con la cual mediante línea de comandos, puedes ir creando las entidades de forma sencilla. Esto viene muy bien para crear las entidades, pero no puedes realizar las relaciones entre ellas, con lo cual, una vez creadas sólo faltaba relacionarlas. Este proceso crea las entidades en forma de clase, con lo cual, si en algún momento quieres llevarte la bbdd a otro tipo de bbdd, no va a haber ningún problema.

Esta herramienta es muy potente, ya que conforme se van creando los atributos de las entidades, se les puede asignar un tipo de dato. Por ejemplo, a un campo email le asignaremos tipo email, a un campo entero, le asignaremos un campo de tipo int, a un campo cadena, le asignaremos un campo de tipo String. Symfony ofrece estas facilidades porque luego a la hora de insertar datos en formularios los valida automáticamente si son correctos o no en base al tipo de dato asignado. De esta forma se evitan numerosas funciones al no tener que validarlos a mano.

Una vez creada, comencé a hacer la parte del Backend y Frontend. Todo iba bien hasta que llegue a la parte de las ventas. Conforme iba viendo cómo se insertaba la información, comenzaron a surgirme dudas de como almacenar los diferentes datos, me surgían casos distintos los cuales teníamos que tener en cuenta, y todos ellos requerían una modificación por mínima que fuese de la base de datos.

Algunas de las modificaciones que realicé fueron las siguientes:

- Desaparecen tablas Autores y Agentes, y se unifica todo en una sola tabla usuarios, ya que los datos a guardaran serán los finalmente iguales, lo único que variarán serán las relaciones en cuanto a sus libros. Esta opción ya la propuse en un principio, pero hasta que no avance no pude concretarlo.

- Desaparece campo función de usuario. De esta forma lo que se crearan serán tablas intermedias entre usuario y libros para saber que libros tiene ese usuario con cada tipo de rol. Si algún usuario no es por ejemplo Agente, pues como agente no poseerá libros. Se crearán las tablas: autores_libros, ilustradores_libros, traductores_libros, agentes_libros y directores_libros. Según tengan libros en estas tablas lo que significará será que para cada venta, obtendrán un % u otro en base a este rol.

- Necesidad de añadir la tabla Importaciones. Después de ir insertando ventas en la bbdd, el compañero encargado de este tema me comentó la posibilidad de que una plataforma después de enviar un informe de ventas, a los días manda otro diciéndote que el anterior es incorrecto, con lo cual surgió la necesidad de eliminar todas las ventas de un fichero ya insertadas. Para este proceso se creó la entidad Importaciones, la cual relaciona un fichero con una venta, de tal forma que si te cargas ese fichero, automáticamente se cargan todas las ventas y posteriormente cargas el fichero nuevo.

Estos ficheros serán almacenados en una carpeta con la fecha para saber cuándo y que ficheros se han subido. En este caso, inicialmente, creé una tabla solo para los ficheros, y una tabla intermedia, donde iba poniendo cada id venta con el id del fichero, finalmente acabe por poner una única tabla importaciones, y añadí un campo a la tabla ventas, donde el valor que introducía era el id de la importación. Esto lo hice así porque me di cuenta de que no me hacía falta una tabla intermedia, y a la hora de realizar borrado en cascada me traía problemas.

- Las ventas que se insertan, pueden ser de países diferentes, estos son almacenados, pero para llevar un control más claro, lo que he realizado es obtener una tabla con todos los códigos de países de tal forma que a cada venta se le asigne el id del país correspondiente.

- Se añade al campo usuarios un atributo que determinará si un usuario es Director de Colección (DC) o no. Si es así, esto significa que tendrá libros como DC que habrá que tenerlos en cuenta a la hora de mostrar las ventas, ya que le pertenecerá un % más.

- Campos is_active, username, roles y salt, a las entidades de usuarios y administradores, ya que si se desea que symfony haga uso de su seguridad se requiere dichos campos.

El diseño final de la base de datos es el que se describe en la Figura 3.5., queda de la siguiente forma:

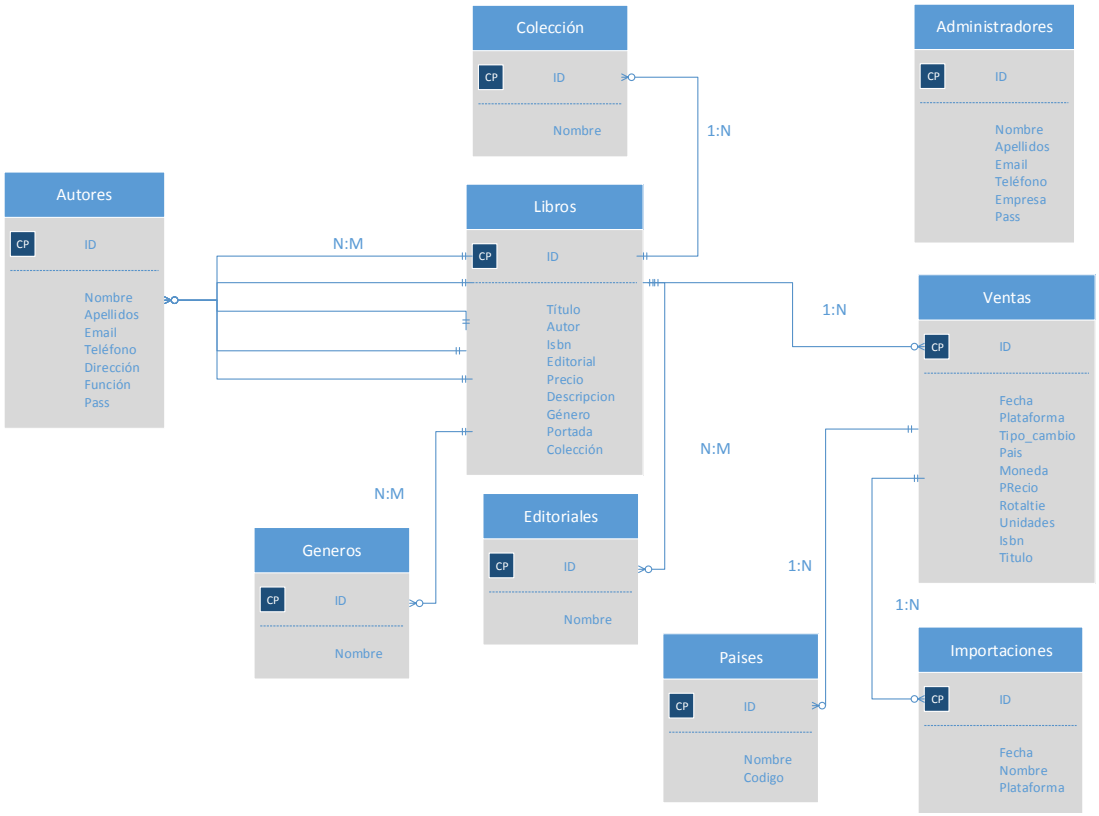


Figura 3.13 Diagrama final de la base de datos.

3.5 Arquitectura del sistema

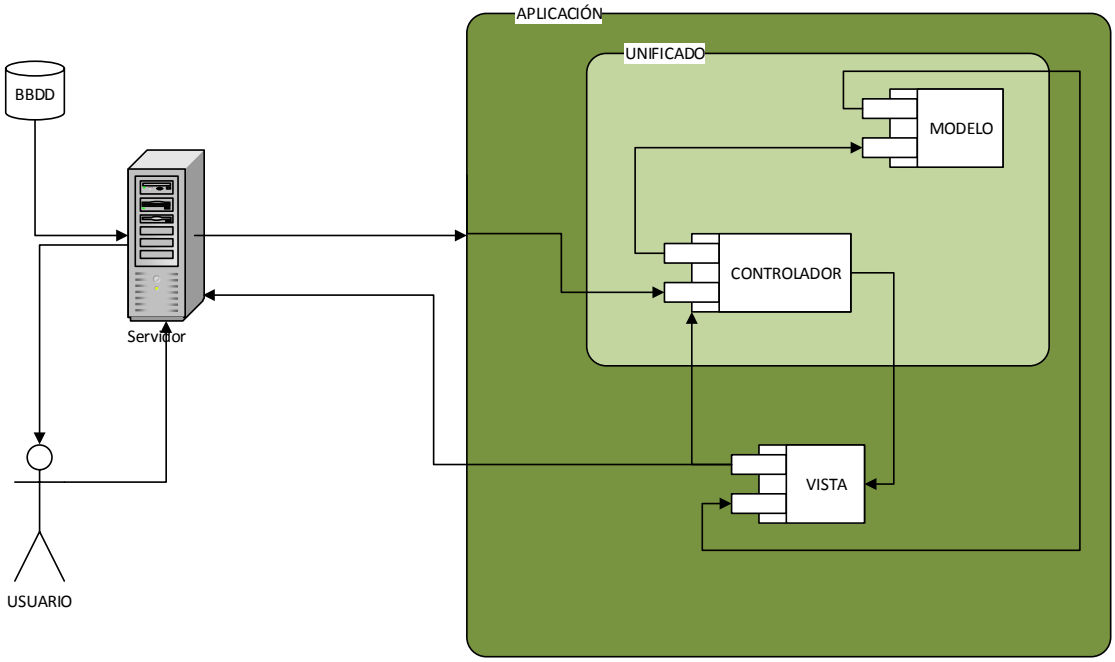


Figura 3.14 Diagrama de la arquitectura utilizada.

En la figura 3.14 muestro la arquitectura que sigue esta aplicación, la cual hace referencia a la de Modelo, Vista, Controlador. En mi aplicación, el modelo y el controlador están relacionados.

Como se puede ver, la aplicación está ubicada en un servidor Apache de mí máquina, este posee una base de datos MySQL donde estarán almacenadas todas las tablas necesarias para que la aplicación funcione correctamente. El flujo de control que se sigue generalmente es el siguiente:

1- El usuario interactúa con la interfaz de usuario (por ejemplo, el usuario pulsa un botón.)
2- El controlador recibe la petición de la acción solicitada por el usuario. Este gestiona el evento que llega.
3- El controlador accede al modelo (si la petición lo requiere), actualizándolo, modificándolo o consultándolo, según el tipo de petición solicitada por el usuario.
4- El controlador manda a la vista las órdenes para modificar la interfaz de usuario. La vista a su vez obtiene los datos del modelo (si se requieren) para generar la interfaz apropiada para el usuario, de tal forma que se muestre la información pedida por la petición.
5- La interfaz del usuario espera nuevas interacciones, comenzando el ciclo nuevamente.

4.- IMPLEMENTACIÓN

En este capítulo se explicarán las herramientas con las que se ha llevado a cabo la aplicación, cómo es la implementación para la funcionalidad, las pruebas realizadas y un presupuesto real.

4.1 Herramientas para el desarrollo

4.1.1 Sistema operativo

El desarrollo de esta aplicación lo he realizado en dos sistemas operativos distintos. Mientras trabajaba en la empresa, utilizaba Linux, con la versión Mint 16. La gran ventaja de utilizar este sistema operativo es que se distribuye como software libre. Por otro lado esta versión es para usuarios de nivel medio, donde utiliza elementos gráficos para mejorar la experiencia del usuario. No me había tocado antes trabajar con esta versión pero la verdad es que ha sido una buena experiencia.

Por otro lado, en mi casa trabajaba con Windows 8.1 que también ha sido exitoso. Sin embargo, aquí echaba de menos la consola que tiene Linux, ya que para realizar la base de datos y crear las clases, con la consola se podía realizar de forma automática, pero bueno, una vez realizada ya no había que modificar nada, con lo cual era secundario.

4.1.2 Editor de textos

Para la realización del código de la aplicación he utilizado Geany [13] en Linux. Este programa no lo había utilizado antes tampoco y la verdad es que me ha parecido muy buena herramienta para programar. Tiene un menú en el cual puedes seleccionar el tipo de archivo que estás realizando, html, php… y éste reconoce automáticamente las distintas etiquetas que componen el lenguaje, dando facilidades a la hora de escribir, o por ejemplo en html, cuando creas un div, éste te lo cierra automáticamente.

Además, en la parte izquierda del editor, reconoce los diferentes métodos y funciones que posee el archivo en el que estás trabajando, de tal forma que a simple vista puedes ver qué métodos tienes e ir directamente a ellos.

En Windows, para implementar código, la herramienta que he utilizado es Notepad++. Con esta herramienta sí que había trabajado antes y lo seguiré haciendo. Es un editor muy completo y cómodo para trabajar. Al igual que el anterior, también posee la opción de seleccionar el tipo de archivo y éste reconoce las etiquetas del lenguaje seleccionado, de tal forma que resulte más agradable a la vista el ver las partes del código.

Ambos editores de textos tienen las siguientes características:

- Búsqueda y reemplazo de texto usando expresiones regulares.
- Seguimiento de código para PHP, HTML y otros.
- Mantener múltiples documentos abiertos en una ventana.
- Numeración de líneas.

4.1.3 Editor de imágenes

Para la realización de las diferentes imágenes de la aplicación he utilizado la herramienta GIMP2. Este es un programa de edición de imágenes digitales, tanto dibujos como fotografías. Este, al igual que los anteriores, es un software libre y gratuito. Además, está disponible para distintos sistemas operativos, como Unix, Linux, Windows, Mac…

GIMP permite el tratado de imágenes en capas, para poder modificar cada objeto de la imagen de forma totalmente independiente a las demás capas en la imagen, la imagen final puede guardarse en el formato xcf de GIMP que soporta capas, o en un formato plano sin capas, que puede ser: png, gif, jpg, etc.

GIMP cuenta con muchas herramientas, entre ellas se encuentran las siguientes;

- Herramientas de selección (rectangular, esférica, por color)
- Herramientas de pintado como pincel, brocha, relleno, texturas, degradados, etc.
- Herramientas de modificación de escala, de inclinación, de deformación, etc.
- Herramienta de manipulación de texto.
- Posee también muchas herramientas o filtros para la manipulación de los colores y el aspecto de las imágenes, como enfoque y desenfoque, eliminación o adición de manchas, sombras, mapeado de colores, etc.
- También posee un menú con un catálogo de efectos y tratamientos de las imágenes.

Con esta herramienta he realizado las distintas imágenes de mi aplicación. Como el logo, las imágenes para la explicación de qué realizar en la web, las imágenes de los menús… Todas ellas han sido retocadas con dicho programa, por ejemplo, para dar transparencias, cambiar tamaños, modificar parte de las imágenes, eliminar cosas y añadir otras, etc.

4.1.4 Editor de hojas de cálculo

Para la generación de los informes de las ventas, he utilizado las hojas de cálculo de Libre Office [14] en la empresa y Excel en mi casa.
Libre Office es una herramienta gratuita y potente con la que, después de haber trabajado con ella, puedo decir que ha sido un éxito en todos los sentidos.

Las funciones de PHPExcel [15], funcionan en los dos por igual, aunque he de decir que para mi gusto, las hojas de cálculo generadas por Libre Office son más adecuadas. Por ejemplo, el darle un ancho automático a las celdas, en Libre Office lo hace bien, sin embargo en Microsoft Excel no.

4.1.5 Navegadores

Como ya comento a lo largo de la memoria, los navegadores que he utilizado principalmente para la ejecución de la aplicación son Mozilla Firefox y Google Chrome. A día de hoy son los navegadores más utilizados debido a que son más rápidos y consumen menos recursos del pc.

También he probado la aplicación en otras como Internet Explorer, pero debido a un problema en cuanto a ejecutar código JS no lo he utilizad tanto. Esto podría ser un problema en cuanto a los usuarios, ya que estos podrían utilizar solo IE, pero se solucionaría en un futuro.

4.2 Funcionamiento

En esta sección voy a detallar el funcionamiento de las partas más significativas de mi aplicación, ya que para cualquier persona con mínimos conocimientos sobre aplicaciones web muchas partes se pueden obviar.

En primer lugar, lo que se necesita es descargarse Symfony. Me descargué la versión 2.4. Una vez que se descarga solo hay que descomprimirla y ya tenemos todos los ficheros necesarios para utilizar este framework.

4.2.1 Estructura de carpetas

La Figura 4.1 muestra la estructura de mi proyecto, donde iré describiendo cada carpeta para tener una ligera idea de qué es cada una.

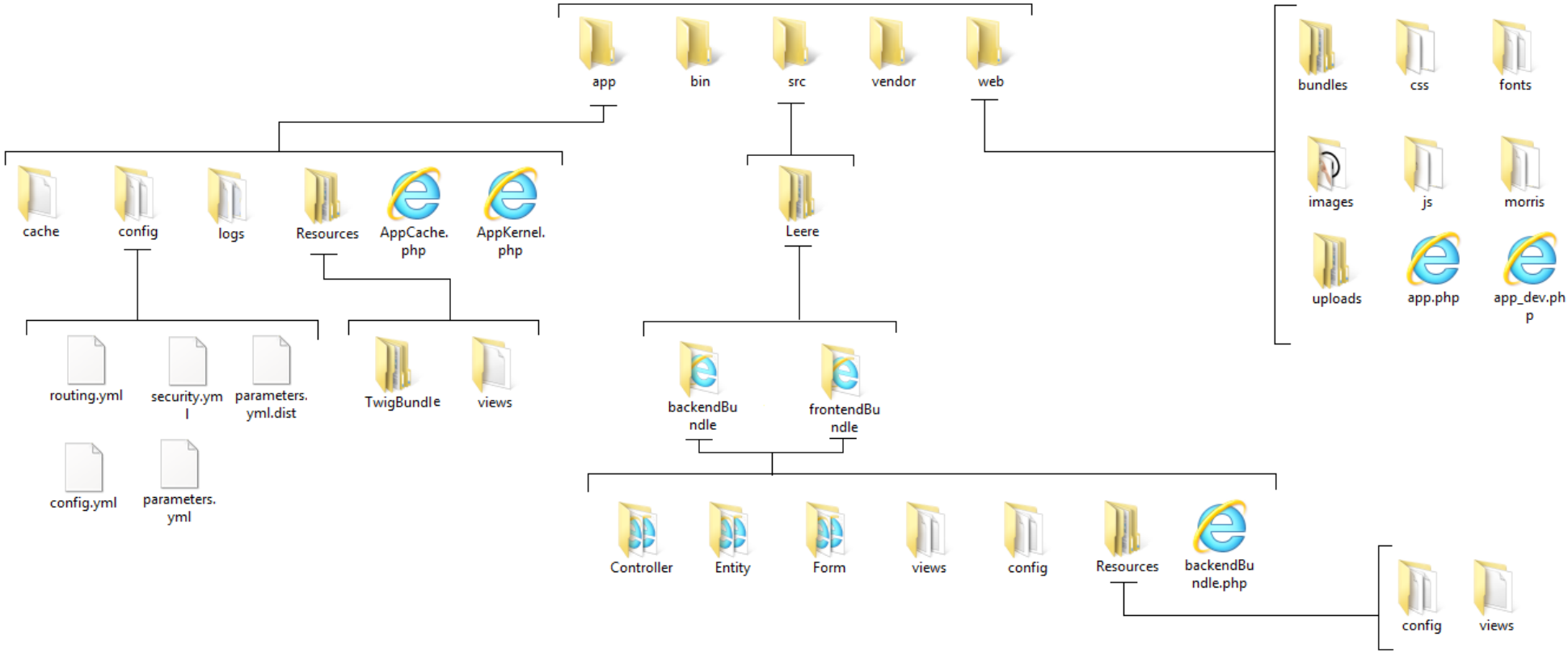


Figura 4.1 Estructura de carpetas de la aplicación.

En el directorio raíz se encuentran cinco carpetas, las cuales son importantes para cualquier proyecto realizado con symfony:

- App: directorio en el cual se encuentran los archivos de configuración de la aplicación. Además en Resources tenemos la plantilla general de la aplicación, donde todas las demás heredan de esa y la plantilla en caso de error, llamada exception, que también hereda de la principal.

- Src: aquí se almacenan los distintos bundles (la explicación de un bundle la hago en el apartado siguiente 4.2.2). En mi caso tengo dos, frontend y backend, ambos poseen una estructura de carpetas interna similar. Dentro de estos tenemos:

    - Controller: directorio donde se almacenan los controladores de cada parte de la aplicación.
    - Entity: donde están almacenadas las clases de las entidades.
    - Form: archivos php para la creación de plantillas automáticamente mediante herramientas de symfony.
    - Resources: aquí dentro existen las plantillas de las vistas de la aplicación y los ficheros de configuración del bundle. (como el routing: utilizado para comunicar la vista con el controlador)

- Vendor: aquí se almacenan las librerías que va a utilizar Symfony como:
    - PHPExcel
    - Twig (gestor de plantillas)
    - Doctrine
    - Composer
- Web: En este directorio se ubican todas las carpetas relacionadas con el diseño de la aplicación, como por ejemplo:
    - Css: archivos de css.
    - Images: imágenes utilizadas por la aplicación.
    - Js: archivos javascript.
    - Morris: librería js para las gráficas.
    - Uploads: ubicación de informes subidos, portadas…

Una vez ya sabemos la estructura del proyecto voy a pasar a explicar las diferentes partes de la aplicación.

Para explicar estas partes, voy a seguir el mismo proceso que en las demás secciones, comenzando por la elaboración de la bbdd hasta la representación de los datos.

4.2.2 Creación de la BBDD

En primer lugar, la creación de la base de datos, como he comentado anteriormente, la he realizado con doctrine, donde a través de la consola de Linux, y mediante los comandos necesarios se crean las clases automáticamente. Para realizar este proceso, primero voy a

explicar qué es un 'bundle' en Symfony, ya que para la creación del modelo es necesario asignarle un bundle.

Un *bundle* (que se podría traducir como *"paquete"*), es como un *plugin*, con la salvedad de que todo el código de tu aplicación se almacena dentro de bundles. No es más que un directorio que almacena todo lo relacionado con una función específica, incluyendo clases PHP, configuración, e incluso hojas de estilo y archivos de *Javascript*.

La creación de éste se hace con el siguiente comando:

php app/console generate:bundle --namespace=Leere/fontend --format=yml

Con esto ya tenemos creado el bundle, donde irá lo referente a la parte de los usuarios. Para la parte de los administradores se realizará la misma operación pero con el nombre backend. Con estas dos líneas, Symfony ya configura automáticamente sus ficheros internos, creando las rutas en los ficheros de configuración y las carpetas por defecto que van a utilizar los bundles, como por ejemplo:

- Entity: para las clases de las entidades.
- Resources: para las plantillas de las vistas y archivos de configuración.
- Controller: para los diferentes controladores que van a existir.

Una vez creado los bundles, ya podemos crear la base de datos que hará referencia a estos.

- Creación de la base de datos: php app/console doctrine:database:create

Para la creación de las entidades el comando a utilizar es el siguiente:

Php app/console doctrine:generate:entity

Una vez introducido este comando, doctrine nos pide una serie de valores como:

- el bundle donde se va a ubicar

- el nombre de la clase

- nombre de campos con su tipo. Es importante darle el tipo correcto, porque luego a la hora de validar los formularios será con lo que haga la validación.

Ejemplo de cómo genera las restricciones para los tipos de datos:

```
/**
 * @var string
 *
 * @ORM\Column(name="email", type="string", length=255, unique=true)
 * @Assert\NotBlank() // examina que no sea nulo
 * @Assert\Email() //tipo de campo email, examina que tenga la estructura de un correo
 */
private $email;
```

Una vez generadas las clases, el siguiente paso es generar el esquema, es decir, volcar cada entidad creada a la base de datos existente, ya que si solo creamos las entidades pero no las volcamos a la bbdd, esta estará vacía.

php app/console doctrine:schema:create

Ahora ya tenemos la base de datos con las clases creadas, las cuales se pueden ver desde mysql.

Para que Doctrine nos cree los Set y Get automáticamente hay que ejecutar la siguiente línea:

Php app/console doctrine:generate:entities (nombre del bundle)

Así ya tenemos las clases creadas, las cuales se pueden ver sin problemas, pero el problema que hay es que estas no están relacionadas. Con lo que el siguiente paso es realizar las relaciones. A continuación voy a poner un ejemplo de una relación N:M para que se vea cómo es el proceso.

En la clase usuarios, creo la siguiente relación, que dará lugar a la tabla intermedia de los libros que posee un agente, cogiendo como claves primarias, el id del usuario y el id del libro:

```
/**
 * @ManyToMany(targetEntity="libros")
 * @JoinTable(name="agentes_libros",
 *     joinColumns={@JoinColumn(name="usuario_id", referencedColumnName="id")},
 *     inverseJoinColumns={@JoinColumn(name="libro_id", referencedColumnName="id")}
 *     )
 */
private $agente;
```

4.2.3 Archivo seguridad

Una vez que ya estaba construido el modelo de datos, el siguiente paso fue comenzar a realizar la aplicación.

Inicialmente comencé por el login, el cual requería el archivo de configuración, a continuación muestro algunas líneas de este para saber su estructura:

```
security:
  firewalls:
    secured_area: (firewall para la parte de administradores)
      *
      *
    secured_area_user: (firewall para la parte de usuarios)
      *
      *
  access_control: (control de acceso a cada parte de la aplicación, controlado mediante roles)
    - { path: ^/(admin|user)/login, roles: IS_AUTHENTICATED_ANONYMOUSLY }
    - { path: /admin/*    , roles: 'ROLE_ADMIN' }
    - { path: /user/*   , roles: 'ROLE_USER' }

  providers: (donde se cogen las diferentes entidades)
    administradores:
    usuarios:
```

encoders: (algoritmos de encriptamiento para encriptar las contraseñas)

4.2.4 Administración de la BBDD

Una vez hecho estas partes, lo siguiente a realizar era la administración de la base de datos, la cual una vez conseguido el funcionamiento para una tabla, el resto era similar, donde las variaciones radicaban en los diferentes tipos de relaciones que un unían cada tabla.

Los pasos de estos son los siguientes:

- En el archivo de configuración de rutas, definir una ruta nueva para que esta vista interactúe con el controlador.
- Crear la función en el controlador correspondiente.
- Crear la vista para el formulario con la extensión: .html.twig

Ejemplo del archivo de rutas para dar de alta a un libro:

backend_registro_libro: //nombre de la ruta
pattern: /menu/registroLibro //nombre que aparece en al url, debe ser unico
defaults: { _controller: backendBundle:funcionesLibro:registroLibro } //en esta línea hace referencia a que bundle se refiere, el nombre del controlador y el nombre del método dentro del controlador.

Ejemplo de controlador para dar de alta un libro:

```
public function registroLibroAction()
  {
    $peticion =$this->getRequest();
    $libro = new libros();
    $formulario = $this->createForm(new crearLibroType(), $libro);
    // load form
    if($peticion->getMethod()=='POST') {
        // notificar al libro que todo ha salido bien
        $formulario->bind($peticion);//asocia el objeto formulario con los datos enviados en peticion
        if ($formulario->isValid()){
          $em = $this->getDoctrine()->getEntityManager();
          $em->persist($libro);//asegurarnos de que los datos son correctos
          $em->flush(); //esta operación es la que hace la inserción en la bbdd
          return $this->redirect($this->generateUrl('backend_menu_libro'));//volver al menu
        }
      }
    Return $this->render('backendBundle:FuncionesLibro:registroLibro.html.twig',array('formulario' => $formulario-> createView()));//hacer referencia a la plantilla de la cual coge los datos.
  }
```

En la vista de la plantilla, para sustituir partes de la vista principal de la que heredan se utiliza {% block nombre_de_la_parte %} {% endblock %} con estas dos etiquetas lo que estamos haciendo es reemplazar lo que tenga dentro la vista principal por lo nuevo (vista de la plantilla). Si en esta nueva ponemos parent() lo que hace es añadir a la principal lo nuevo, conservando las dos cosas.

Ejemplo, en la plantilla de la vista del menú principal de los usuarios, aparece un carrusel con imágenes informativas, este carrusel necesita cargar un archivo propio de css para poder mostrarlo. Ese css especifico del carrusel y solo se añadirá en dicha plantilla, para ello se sustituirá el {% block stylesheetsMenu %} en la plantilla del menú principal cargando los css principales mediante el {{ parent }} y el Nuevo css que se necesite. De ésta forma al cambiar de plantilla, este css dejará de cargarse para las demás ya que no es necesario para las otras vistas.

```
{% block stylesheetsMenu %}
    {{ parent() }}  //cargo los css de la pantilla principal
    <link href="{{ asset('css/camera.css') }}" type="text/css" rel="stylesheet" id='camera-
    css' media='all' >  //cargo el Nuevo css
{% endblock %}
```

Para el caso de modificar un registro, los pasos son los mismos, excepto a la hora de realizar la vista que es algo diferente, ya que primero hay que cargar los datos en el controlador para luego pasarlos a la vista y esta los muestre. Para este proceso Symfony utiliza sus formularios especiales, donde mediante un constructor va añadiendo los diferentes campos a dicho formulario. No es obligatorio hacerlo así pero ya que Symfony trabaja de este modo lo he realizado así.

4.2.5 Inserción de ventas

Los detalles de este apartado lo explicaré en el siguiente punto, en las pruebas, ya que al ir realizando las distintas pruebas era donde iban saliendo las distintas modificaciones para realizar adecuadamente las inserciones. En términos generales, esta parte de la aplicación consta de búsquedas en los ficheros de las ventas para ir guardando los datos que nos interesan.

La búsqueda trata de leer el fichero e inicialmente guardarnos las columnas en las que están almacenados los datos que queremos, una vez tenemos las posiciones, lo que hacemos es recorrer el fichero entero para ir tratando los datos y finalmente insertarlos. En este proceso se hará sólo un fflush(); ya que es la operación que más tarda y con hacerlo una vez es suficiente, debido a que va añadiendo los datos al objeto.

Este proceso tiene la ventaja de que se pueden subir varios ficheros de la misma plataforma a la vez. De esta forma se le ahorra al administrador la tarea de tener que subir ficheros de uno en uno.

Para las dos formas de actualizar los datos de las ventas de BN y Apple, como he comentado al inicio de esta memoria sobre el tipo de cambio, hay dos opciones:

6- Insertar el valor del tipo de cambio, seleccionar las fechas y actualizarlo.

7- Mediante un fichero proporcionado por la propia plataforma. Primero el administrador deberá seleccionar el archivo, seguido pondrá las fechas pertenecientes a ese archivo, seguido lo que se realiza es una búsqueda para obtener los distintos tipos de cambio según la moneda que tenga cada venta, y dependiendo de la moneda de cada venta esta se actualizará con un tipo de cambio u otro.

8- Existe otra forma de insertar el tipo de cambio, es un proceso automático ya que los datos del tipo de cambio son proporcionados junto con las ventas en un mismo archivo (esto ocurre en Amazon). En este caso lo que hago es un doble recorrido del archivo, uno para insertar las ventas y acto seguido otro recorrido para obtener los datos del tipo de cambio y actualizar las mismas. Este proceso puede ser de dos formas:

   - El tipo de cambio se da en una sola línea, con lo que se toma ese tipo de cambio y se aplica.
   - El tipo de cambio viene en muchas líneas: esto conlleva a leer todas y realizar una media de todas, el valor obtenido de la media será el tipo de cambio que se aplique.

Además, para la inserción de las ventas de Amazon, nos da un fichero de ventas por cada país, y son todos distintos menos los de Europa. Debido a esto y al doble recorrido, llevó más tiempo su implementación.

4.2.6 Función buscador libros

Para la parte de los usuarios existe un buscador en donde éstos pueden consultar sus libros, y se les da la opción de visualizar la gráfica del estado de las ventas. Para esta implementación el proceso es el mismo que para la administración de los datos, lo único que cambia es que el controlador hace una búsqueda en la tabla libros, por el ISBN, título, descripción y autor, y devuelve en una tabla los libros similares que encuentra.

```
$palabrasClave = $_POST["searchTodo"]; //para hacer referencia al campo de la vista

    $datosBuscar=$em->createQuery("SELECT v FROM frontendBundle:libros v WHERE v.titulo
LIKE :titulo OR v.autor LIKE :autor OR v.isbn LIKE :isbn OR v.descripcion LIKE :descripcion")

            ->setParameter('titulo', '%'.$palabrasClave.'%')

            ->setParameter('autor', '%'.$palabrasClave.'%')

            ->setParameter('isbn', '%'.$palabrasClave.'%')

            ->setParameter('descripcion', '%'.$palabrasClave.'%')->getResult();
```

4.2.7 Generación de informes

Para la generación automática de informes, he utilizado la librería de PHPExcel [15]. Para poder utilizar esta librería tuve bastantes problemas, ya que no conseguía añadirla a Symfony. Finalmente conseguí insertarla de la siguiente forma:

1- Descargar bundle de Excel, el cual ya contiene la librería.
2- Añadir al archivo composer.json la siguiente línea: "liuggio/excelbundle: 2.0"
3- Actualizar el composer: sudo composer self-update
4- Añadir el nombre del bundle al archivo kernel.conf
5- Borrar la cache: sudo rm –rf app/cache/*

Estos fueron los pasos que seguí para poder utilizar dicha librería. En el paso uno, me descargué de una web el bundle: [16], el cual contiene los ficheros necesarios.

Seguido añado dicha línea al archivo composer, de tal forma que al realizar a continuación el update, Symfony encuentra algo nuevo y actualiza sus ficheros de configuración para que se pueda utilizar dicho bundle. Una vez tenemos estos tres primeros pasos lo que hay que hacer es decirle al kernel el directorio dónde está dicho bundle ya que este parámetro no lo configura. Me di cuenta porque Symfony da un error indicando que falta definir el bundle en el kernel.

Y por último, es aconsejable borrar la cache, por el rastro que pueda haber, ya que antes no se sabía de su existencia.

Para poder utilizar las funciones de una librería externa a Symfony, es necesario añadir delante de cada función el símbolo " \ " ya que estas funciones son ajenas a Symfony y no son propias de él, con lo cual no están definidas, y para hacer referencia a ellas es necesario hacer uso de la " \ ".

En primer lugar, para ver cómo funcionaba la librería realicé multitud de ejemplos para ver cómo se iban viendo las cosas. Una vez entendí el funcionamiento, lo primero en hacer fue la cabecera, donde se muestra el logo de la empresa, con los datos de esta, y un título para saber de qué es el informe.

Los informes que generaban ellos no tenían este encabezado, pero se me ocurrió ponerlo para simular una especie de "factura" a los clientes.

El siguiente paso fue obtener los datos de las ventas de éste, según el periodo que seleccionase y con qué función. Para el caso de seleccionar todas las funciones, la aplicación generará una hoja para cada uno. De esta forma no se mezclaran los datos y se mostrarán los precios correspondientes a cada uno.

Este código está bastante bien explicado en el código del controlador que realiza esto. Debido a que es un código muy extenso, no he visto apropiado plasmarlo todo aquí, ya que todas son interesantes y necesarias para la creación de informes. Para ello muestro algunas de las funciones que realiza dicho código, estas son:

- Poner el logo.
- Poner las celdas con tamaño automático.
- Dar formato a las celdas.
- Hacer las consultas para obtener los datos.
- Como ir trabajando con diferentes páginas dentro de un mismo Excel.
- Al encontrar mismos libros, meter el número de unidades en su celda y no repetir filas. con los mismos libros.
- Seleccionar fechas.

4.2.8 Generación de gráficas

Para la generación de gráficas, lo primero que hice fue visualizar otras aplicaciones como la de Apple para coger alguna idea sobre cómo realizar esta parte.

La siguiente cuestión era buscar la mejor librería para desarrollar esto. Tras probar diferentes como: JpGraph, pChart… me decidí por la librería Morris [17] de JavaScript. Me decanté por ésta por su sencillez a la hora de representar las gráficas, ya que son muy claras y dan la información necesaria. Además, no me llevó mucho tiempo aprender a utilizarla, debido a que su uso es sencillo e intuitivo. Esta librería posee ejemplos sobre diferentes tipos de gráficas, las cuales se elegirán según nos convenga.

Para realizar esta implementación, lo primero que generé fue el menú, con las diferentes opciones a elegir para ver diferentes datos.

- Por un lado se podrá hacer un seguimiento de todas las ventas juntas, mes a mes, ya sea por año o por fecha.
- Por otro lado, se podrá realizar un seguimiento de las ventas del mes anterior viendo todos los libros que posee el usuario.
- Por otro, se podrá seleccionar el libro deseado, y ver las ventas de este al cabo de un año o de la fecha seleccionada.
- Además se podrá ver en que plataformas se vende más, mostrando un porcentaje.

Para esta función de la aplicación en donde se generan gráficas para ver las ventas, se podría haber planteado de otra forma a como lo hemos planteado. Otra idea interesante sería incrustar la creación de todas las gráficas en una misma vista, de esta forma hubiese ahorrado trabajo en evitar crear múltiples vistas. Pero uno de los requisitos que determinamos desde el principio fue garantizar la inteligibilidad para cualquier usuario. Por ello, opté por hacerlo en diferentes menús, de esta forma el usuario podrá saber en todo momento que va a generar.

El controlador encargado de crear las gráficas, posee numerosas funciones para la creación de estas, las cuales algunas son similares. Para estas, podría haber realizado una única función y haberla invocado en ocasiones similares para generar gráficas, pero debido a la cantidad de instrucciones que realizo, decidí implementar una por tipo de gráfica y así evitar errores, ya que al utilizar una función para creaciones distintas puede acarrear problemas en cuanto a compatibilidad.

El proceso es similar al seguido en la creación de los informes. Pasos:

1- Obtener el usuario con el cual se ha iniciado sesión.
2- Obtengo los libros de ese usuario según el rol seleccionado.
3- Obtengo los datos de los libros.
4- Obtengo las ventas de esos libros según las fechas seleccionadas.

En el caso de seleccionar un rango de fechas para obtener las ventas, hay que pasarle las fechas correctamente a la consulta, con lo cual voy mirando cuál es el rango introducido y según los meses seleccionados se irán dando unos valores a la fecha u otros, ya que las consultas de las ventas se hacen por meses.

Por ejemplo:

- Si la fecha introducida es del 01-01-2014 al 31-01-2014, se le dará:

  - el mes enero desde el 01 al 31

- Si la fecha introducida es del 02-01-2014 al 03-02-2014, se le dará:

  - el mes enero desde el día 02 al 31

  - el mes febrero desde el 01 al 03

- Si la fecha introducida es del 02-01-2014 al 15-04-2014, se le dará:

  - el mes enero desde el día 02 al 31

  - el mes febrero desde el 01 al 28

  - el mes marzo desde el 01 al 31

  - el mes abril desde el 01 al 15

Para obtener las ventas utilizo por ejemplo la siguiente consulta a la base de datos:

```
$ventasObtenidas = $em->createQuery("SELECT sum(v.unidades) AS unidades FROM
frontendBundle:ventas v WHERE v.libro= :libro_id AND v.fecha >= :fecha_inicio AND v.fecha <=
:fecha_fin GROUP BY v.titulo")->setParameter('fecha_inicio', $dateInicio)-
>setParameter('fecha_fin', $dateFin)->setParameter('libro_id', $id)->getResult();
```

En la cual hago un sumatorio de las unidades del libro seleccionado, con la condición de que las ventas estén entre un rango de fechas.

Una vez obtengo las ventas de los libros, lo que hago es pasarle a la vista los datos obtenidos de la siguiente forma:

```
return $this->render('frontendBundle:FuncionesVentasGraficas:listLibrosGrafica.html.twig',
array('libros' => $datosLibros, 'unidades' =>
$ventas_totales,'nombreGrafica'=>$nombreGrafica));
```

Con esta instrucción, lo que hago es llamar a la vista listLibrosGrafica.html.twig, la cual se ubica en la carpeta FuncionesVentasGraficas del bundle frontendBundle, y le paso un array

con las unidades de las ventas, otro con los nombres de los libros y otro con el nombre que se le dará a la gráfica.

Por último, sólo queda interpretar estos datos en la vista, en la cual recorro los arrays con JS ya que la librería utiliza JS, y se los paso a la función de generar gráfica la cual realizará la representación de éstos.

4.3- PRUEBAS

En este punto de la memoria, voy a describir las pruebas que he ido realizando para la realización de cada parte de la aplicación. Como ya he comentado anteriormente, he ido realizando el proyecto en distintas partes, las cuales voy a detallar las pruebas pertinentes que he realizado en cada una de ellas.

Para cada una de las partes de la aplicación, iba realizando las pruebas oportunas para comprobar que el funcionamiento era el correcto. Además, para la vista del frontend, también realizaba mi compañero alguna prueba, para dar su opinión y ver si en algún lado fallaba.

Al realizar una aplicación web, una de las características que definen a este tipo de proyectos, es que gracias a su lenguaje, te permite ver en todo momento las modificaciones realizadas.

Debido a eso, se hacen pruebas constantemente, de tal forma que te cerciores de que lo que estás haciendo está funcionando como quieres y de forma correcta.

Las pruebas que he ido realizando en cada apartado son las siguientes:

- Modelo relacional: para esta parte, las pruebas que realicé fueron una vez que ya tenía la parte del backend hecha, iba insertando /actualizando/eliminando datos para ver que ésta estuviese funcionando correctamente. Como en todo proyecto, siempre se te suele colar alguna relación mal hecha o un tipo de una variable incorrecto o algo del estilo, pero nada grave que no se pueda solucionar con simples cambios.

- En la realización del backend, primero comencé con la realización del login, el cual me dio problemas porque tuve que añadir, como ya he dicho anteriormente, unos atributos que necesitaba Symfony para dar seguridad a la aplicación. Una vez hecho esto, lo siguiente era crear los diferentes formularios para la inserción/actualización/eliminación, y ver que estos validaban los datos y mostraban los datos correctamente. Aquí, a la hora de insertar datos de los libros, cuando éstos iban relacionados con otras tablas, me costó entender cómo trabajaba symfony con este tipo de datos, pero una vez solucionado, para las demás era similar.

- Para la inserción de las ventas, es uno de los apartados que más pruebas realicé debido a la gran cantidad de datos distintos que tenía y los cuales me tenía que asegurar de que estos eran correctos a la hora de introducirlos, ya que son datos de ventas y a un cliente no podemos mostrarle cosas que no ha vendido o viceversa.

  En un primer momento, comencé a trabajar los ficheros de las ventas con PHPExcel, lo cual no resulto muy satisfactorio. Tras observar la estructura de los documentos y realizar sucesivas pruebas, llegué a la conclusión de que esos informes se podrían guardar de otra forma, para así tratarlos mediante simples búsquedas.

Tras trastear con las aplicaciones internas de las plataformas, nos dimos cuenta que se podían descargar los ficheros como .csv lo cual facilitaba mucho las cosas, y era lo que iba buscando. De esta forma es como trabajar con un archivo de texto.

Lo que realizo a la hora de tratarlos es un recorrido por el archivo de tal forma que voy guardando los datos necesarios.

El responsable de la empresa, me iba diciendo qué datos eran los que nos interesaban y los que correspondían con nuestra base de datos. Al proceder de archivos .csv algunos datos no se podrían guardar en la bbdd tal cual, y requerían una trasformación.

Los ficheros correspondientes a Google, necesitaban un doble recorrido, ya que son totalmente distintos a los demás. Estos aportan el tipo de cambio en otra tabla, la cual es necesaria para rellenar todos los datos correctamente.

Ejemplos de datos que he ido tratando:

- Campo fecha, los valores de fecha te los podías encontrar de cualquier forma, el mes primero o el año o el día depende, por otro, lado en alguna plataforma el mes no es correcto y tienes que poner el mes anterior.
- Los valores referentes a los precio no se pueden insertar tal cual, ya que mysql no admite las comas, hay que cambiarlas por puntos.
- Los campos unidades, cada plataforma los interpreta como quiere, en unas las unidades devueltas las pone como -1 en otras como 1, con lo cual hay que tener en cuenta para hacer el total de unidades vendidas.
- Campo país, realizo una búsqueda en una tabla países para ver qué país es.

- Seguido realicé la parte del frontend. Para realizar ésta aproveché lo que ya tenía del backend y lo adapté de tal forma que hiciese referencia a los usuarios. Aquí las pruebas fueron en principio mínimas, ya que sólo probé si funcionaban los diferentes menús y a la hora de mostrar los datos de la empresa. En este apartado, realicé el buscador de libros, donde no me dio ningún problema.

- Acto seguido realicé la parte de la generación de los informes. Aquí, al igual que en el apartado de insertar ventas, realicé multitud de pruebas para cerciorarme de que los datos eran correctos. Esto suponía más tiempo de prueba, ya que había que seleccionar los diferentes filtros, abrir el informe y hacer las comprobaciones.

En este apartado sí que utilicé las librerías de PHPExcel, primero tuve que entender cómo funcionaba y luego, en base a pruebas, ir viendo lo que hacía cada función.

El mayor problema en esta parte de la aplicación era el obtener los datos de diferentes tablas, ya que el proceso a seguir es ir pasando por cada tabla e ir guardando datos según se relacione con la anterior, en vez de hacer una única consulta para obtener todo, que sería lo más fácil. Mi idea era hacer una única consulta, pero el informático de Leer-e me recomendó lo primero debido a que me iba a dar menos problemas.

Primero obtengo el usuario conectado, seguido obtengo los libros que posee dicho usuario, según sus roles o si el usuario lo desea con todos las roles de éste. Acto seguido obtener los datos de los libros, y por último, los datos referentes a las ventas.

Primero realicé pruebas para el caso de elegir un rol (agente, autor, traductor, ilustrador o director) y mostraba los resultados en una hoja con un tipo de formato similar al de una factura.

Una vez que esto funcionaba, realicé lo mismo pero para el caso de que el usuario desee el informe de todos los roles. Aquí muestro un único documento pero la información se divide en diferentes hojas, de tal forma que no se mezclen los datos, asignando cada hoja a un rol de éste. Si el usuario no tiene libros con alguno de los roles (escritor, ilustrador, traductor, agente, director), esta hoja no se le creará.

-Después de realizar los informes correctamente, me puse a representar la información de las ventas mediante gráficas. Este apartado también requería multitud de pruebas. Pruebas como:

- comprobar si los datos se están cogiendo bien.

- comprobar si los datos se pasan del controlador a la vista correctamente.

- comprobar que los datos no se repitiesen.

Además, para realizar estas gráficas, utilicé la librería Morris. Es una librería en Javascript, la cual facilita la implementación de gráficas para representar datos. Primero, tuve que ver cómo trabajaba viendo ejemplos y realizando cambios sobre los mismos para entender el funcionamiento. Una vez que dominaba los ejemplos, lo siguiente era crear dichas gráficas pero adquiriendo los datos de mi bbdd.

Como ya he dicho antes, tuve que realizar numerosas pruebas, ya que había que cerciorarse de que los datos eran los correctos y éstos no se repitiesen.

Además de las pruebas expuestas, también realice pruebas en distintas máquinas para ver que el funcionamiento y la visualización era la misma.

También he probado la aplicación en diferentes navegadores y el interfaz es similar. Dependiendo del navegador puede cambiar algún detalle cómo:

- las notificaciones de datos mal introducidos,

- las gráficas generadas.

En conclusión, pequeños detalles que no marcan diferencia, simplemente muestran los errores o los formularios en su forma predeterminada.

He de añadir, que al abrir la web con Internet Explorer, todo lo relacionado con los scripts de las gráficas no se muestran. Después de informarme sobre este error, he visto, que Internet Explorer tiene muchas restricciones en cuanto a la ejecución de scripts. Este mismo problema lo he tenido en varios proyectos anteriores, donde la aplicación funcionaba perfectamente en otros navegadores y en Internet Explorer no.

Buscando soluciones, he peguntado a mis compañeros y todos hemos coincidido en lo mismo, que este navegador no deja ejecutar ciertos scripts. Para intentar solucionarlo, he intentado activar todas las opciones de configuración que trae IE, (obviamente todas las opciones relacionadas con scripts, estaban deshabilitadas) y sí que hace mención de mostrarla, pero no llega a mostrar las gráficas. Debido a este problema, la aplicación tendrá una restricción, solo se podrá ejecutar en navegadores distintos a IE.

4.4- PRESUPUESTO

En este apartado, voy a cuantificar el coste que le supondría a la empresa mi trabajo realizado durante estos cuatro meses. Esta aplicación ha sido desarrollada por un único programador, y se ha ido implementando cada semana distribuyendo las horas de la siguiente forma:

DÍAS	Lunes a Jueves	Viernes
HORAS	5.5 horas	5 horas

Esta forma de distribuir las horas hacen un total de:

- 5.5 horas * 4días= 22 horas
- 5 horas * 1día= 5 horas
- Total= 22 horas + 5 horas= 27 horas/semana

La duración ha sido de unos 4 meses aproximadamente, para ser exactos, un total de 18 semanas, lo cual implica las siguientes horas empleadas:

18 semanas * 27 horas= 486 horas.

Como ya he comentado que el programador tendría un contrato en prácticas, suponemos que va a cobrar a 6 € la hora, daría un total de:

486 horas * 6€/hora = 2916 €

Una vez obtenido el coste a pagar al programador, la empresa Leer-e tiene que añadir los costes indirectos que le supone que el programador este utilizando las instalaciones y recursos de la empresa.

Habría que descontar los siguientes gastos:

- Gasto energético y gas. (30 €/mes)
- Gasto de agua. (10 €/mes)
- Gastos de material. (folios, tinta, bolígrafos…) (20€/mes)
- Gasto en un seguro al trabajador (seguridad social). (300 €/mes)
- En el caso de que se estropee el pc, reemplazarlo por otro. (800 €)
- Gastos de desplazamiento en el caso de acudir a reuniones. (10 €/mes)

Sumando todos estos gastos, obtendríamos un total de:

Gasto	Tiempo en meses	€ al mes	Total €
Energético y gas	4	30	120
Agua	4	10	40
Material	4	20	80
Seguro	4	300	1200
Pc averiado*		800*	800*
Desplazamientos	4	10	40

1480 euros. Suponiendo que el pc no se avería. Lo que supondría un gasto total para implementar esta aplicación de:

2916 € + 1480 €= 4396 €

Si el pc se averiase el coste incrementaría en 800 €.

Aspectos a tener en cuenta:

- No habría que comprar ninguna licencia de ningún tipo, ya que todo lo utilizado es software libre, con lo cual no requiere ningún gasto.
- La empresa ofrece cursillos gratuitamente para aumentar tus conocimientos, con esto se quiere reflejar, que, te dan la oportunidad de aprender sin ningún coste.

5.- CONCLUSIONES Y LINEAS FUTURAS

En este capítulo se muestran las conclusiones obtenidas y las mejoras que se pueden realizar.

5.1 CONCLUSIONES

Después de hacer este Trabajo Fin de Grado, estoy muy satisfecho por todo el esfuerzo que he realizado para su implementación.

Antes de nada, me gustaría señalar que una vez implementada la aplicación y probada, quizás no quede reflejado todo el trabajo que hay detrás.

En cuanto a funcionalidad, la aplicación no es muy extensa, pero cada uno de los procesos ha llevado un trabajo duro y laborioso. Por otro lado, la aplicación no se ha probado con todos los datos que me hubiese gustado, ya que el encargado de la empresa no me ha facilitado toda la información referente a los libros. Para ello se han utilizado datos de libros insertados por mí, de tal forma que concuerden con los informes de ventas reales, y así ver el funcionamiento de la aplicación.

Con todo ello, realmente se ha logrado transmitir el objetivo al usuario y a la empresa. A la empresa se le ahorrará trabajo, y de esta forma tendrán los datos de las ventas bien almacenados. En cuanto a los usuarios, les resultará más cómodo ver el progreso de sus ventas mediante gráficas.

En cuanto a mi primer contacto con Symfony, al principio fue un reto que me propuse, y tras varias horas de trabajo, no puedo decir que sea un experto, pero creo que me manejo bastante bien con este framework, con ello, en mi próximo trabajo con esta herramienta, estoy seguro de que me irá bastante mejor.

Por otro lado, estoy contento por la mejora que he conseguido en cuanto a programar en los lenguajes utilizados. En cada proyecto que se realiza, se aprenden cosas nuevas, ya que surgen problemas que son específicos de la aplicación. Además para la realización de este proyecto, he utilizado lenguajes nuevos que antes no había usado, y siempre viene bien saber cosas nuevas.

Para concluir, como experiencia, he de decir que he disfrutado mucho realizando este proyecto en la empresa Leer-e, ya que el trato ha sido inmejorable y mi interés ha hecho posible que salieran las cosas adelante. Ha merecido la pena todo el esfuerzo que he hecho y espero que en un tiempo esta web sea utilizada por usuarios de Leer-e.

5.2 LINEAS FUTURAS

Esta aplicación está pensada para reemplazar a la aplicación que utilizaban hasta hace un tiempo para llevar el control de las ventas y libros. Debido a que la utilizada anteriormente es tan compleja, que resulta poco práctica. En mi opinión, en cuanto a diseño no es una aplicación que llame la atención, pero el funcionamiento es el que se necesitaba.

Lo primero me gustaría señalar que si continuase en la empresa, una vez se insertasen todos los datos correctamente, en primero lugar, realizaría un diseño mediante herramientas de diseño de páginas web, de esta forma obtendría una interfaz más atractiva y llamaría la atención de los usuarios.

Una de las funciones que se podría implementar sería un autocompletar, es decir, a la hora de buscar o rellenar campos, que la web te de sugerencias para autocompletar la búsqueda, o rellenar campos de los formularios.

Por otro lado, la empresa, en un futuro quiere que sean los usuarios quienes lleven el control de sus libros, es decir, ellos puedan administrar los datos de sus libros como deseen. Para ello, una vez que realicen estos cambios, habría que realizan una aprobación por parte de la empresa para verificarlos, ya que no se puede permitir que los datos sean incorrectos, porque podría acarrear numerosos problemas.

Una implementación interesante para añadir a la aplicación sería el proceso de publicación de libros. Para publicar a la venta un libro nuevo, hay que seguir un proceso con determinadas fases, hasta ahora tenían el problema de que muchos libros no se sabía en qué fase se encontraban ya que no tenían una aplicación para llevar el control. Para ello, yo les propuse implementar esa funcionalidad en mi aplicación, de tal forma que exista una tabla “librosSinPublicar” e ir viendo en todo momento en que fase se encuentra cada libro, y una vez se hayan cumplido todas las fases, se publique el libro y este pase automáticamente a la tabla “libros”, en donde se encuentran todos los libros de los que dispone la editorial para vender.

Con esta implementación, es una forma de saber cómo progresa la creación de los libros y si se retrasa en alguna fase saber por qué y quien ha sido el culpable ya que los empleados escribirán comentarios.

Otra implementación interesante sería la creación del sitio web traducida a diferentes idiomas, como inglés o francés. Esta idea surge debido a que a día de hoy, numerosas aplicaciones están disponibles en diferentes idiomas, y en mi opinión es algo necesario, porque nunca sabes que prefieren los clientes, para ello es recomendable dar opciones.

Debido a las diversas horas empleadas para tener todos los casos en cuenta en las funciones de insertar y mostrar ventas, no he podido optimizar determinadas funciones de la aplicación. Otro punto de mejora que realizaría sería optimizar esas funciones.

En este momento, los administradores no tienen la funcionalidad de ver las ventas en informes y gráficas como los usuarios. Otra posible implementación sería hacer lo mismo para ellos, donde podrían ver gráficas por usuario, y ver sus ventas a simple vista. Esta implementación no sería muy laboriosa, solo habría que cambiar las consultas realizadas para obtener los datos en las ventas.

6.- BIBLIOGRAFÍA

En este capítulo se muestran algunos de los enlaces consultados.

6.1 Enlaces Consultados

1. Editorial Leer-e, http://www.leere.es [20/06/2014]
2. Herramienta Symfony, http://symfony.es [20/06/2014]
3. Manual Php, http://www.php.net//manual/es/index.php [20/06/2014]
4. Manual Html, http://www.w3schools.com/html/DEFAULT.asp [20/06/2014]
5. Manual Css, http://www.w3schools.com/css/DEFAULT.asp [20/06/2014]
6. Manual Javascript, http://librosweb.es/javascript/ [20/06/2014]
7. Manual Doctrine, http://gitnacho.github.io/symfony-docs-es/book/doctrine.html [20/06/2014]
8. Manual Twig, http://twig.sensiolabs.org/ [20/06/2014]
9. Información sobre MySQL, http://es.wikipedia.org/wiki/MySQL [20/06/2014]
10. Información sobre PostgreSQL, http://es.wikipedia.org/wiki/PostgreSQL [20/06/2014]
11. Información sobre Oracle, http://es.wikipedia.org/wiki/Oracle [20/06/2014]
12. Información sobre SQL, http://es.wikipedia.org/wiki/Microsoft_SQL_Server [20/06/2014]
13. Geany, http://www.geany.org/ [20/06/2014]
14. Libre Office, http://es.libreoffice.org/ [20/06/2014]
15. Manual phpExcel, https://phpexcel.codeplex.com/ [20/06/2014]
16. Bundle phpExcel, https://github.com/liuggio/ExcelBundle [20/06/2014]
17. Librería Morris, http://www.oesmith.co.uk/morris.js/ [20/06/2014]

Otros enlaces consultados para la implementación de la aplicación y los cuales no se mencionan a lo largo de la memoria:

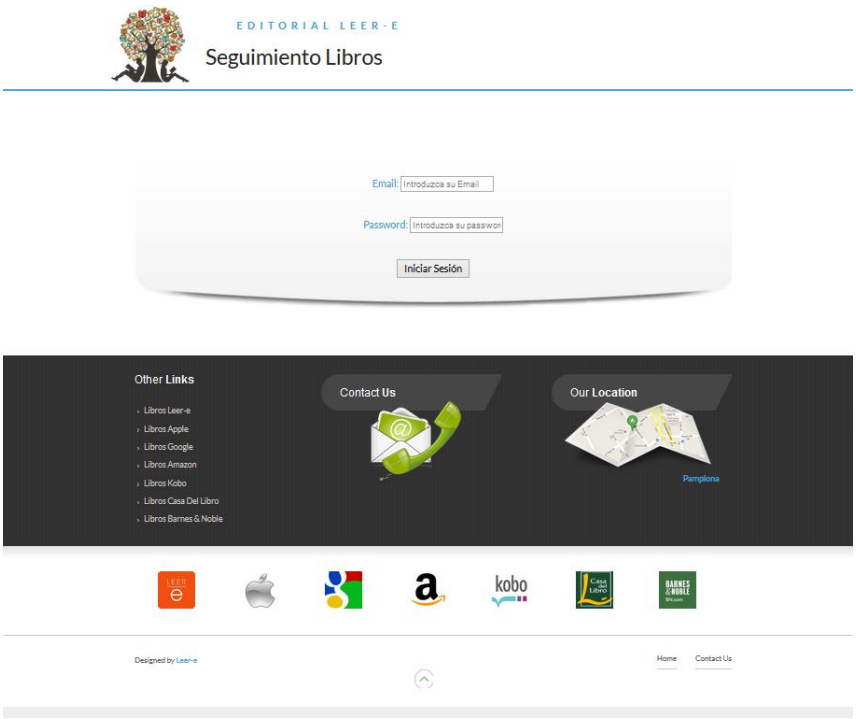
18. Funciones JavaScript, http://librosweb.es/javascript/capitulo_4/funciones.html [20/06/2014]
19. Plantillas Symfony, http://librosweb.es/symfony_2_x/capitulo_7/plantillas.html [20/06/2014]
20. Platillas Twig, http://gitnacho.github.io/Twig/templates.html [20/06/2014]
21. Manual PhpExcel, http://es.scribd.com/doc/54054793/PHPExcel-Documentation-de-Desarrollo [20/06/2014]
22. Configuración archivo seguridad Symfony, http://gitnacho.github.io/symfony-docs-es/book/security.html [20/06/2014]
23. Restricciones en Symfony, http://librosweb.es/symfony_2_x/capitulo_11/restricciones.html [20/06/2014]
24. Información sobre Symfony, http://es.wikipedia.org/wiki/Symfony [20/06/2014]
25. Consultas sobre páginas web, http://www.desarrolloweb.com [20/06/2014]

7.- ANEXO, FUNCIONAMIENTO DE LA APLICACIÓN

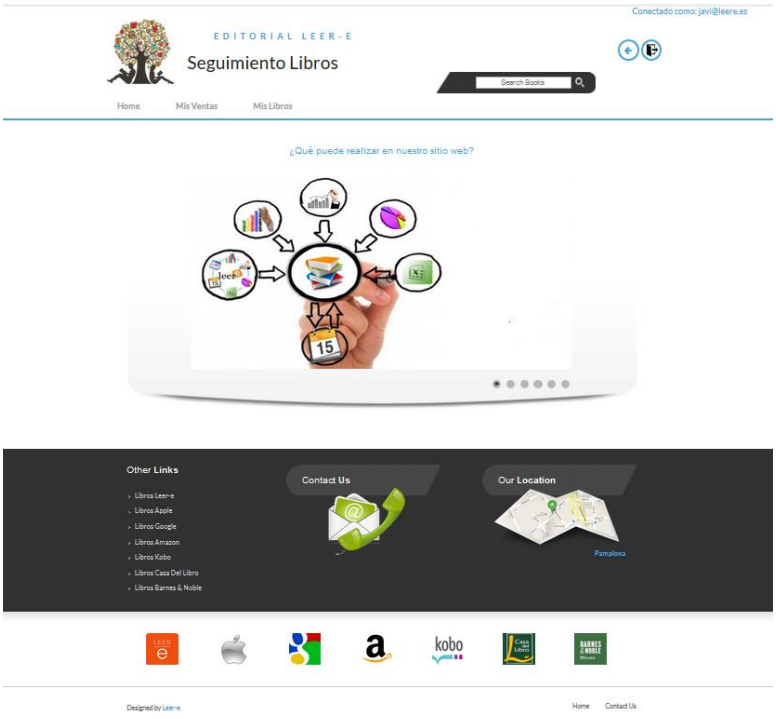
En este capítulo se muestran algunos pantallazos de cómo es la aplicación de los usuarios.

En esta sección, voy a hacer una guía sobre la aplicación para ver el funcionamiento y tener una idea de cómo es la parte del frontend.

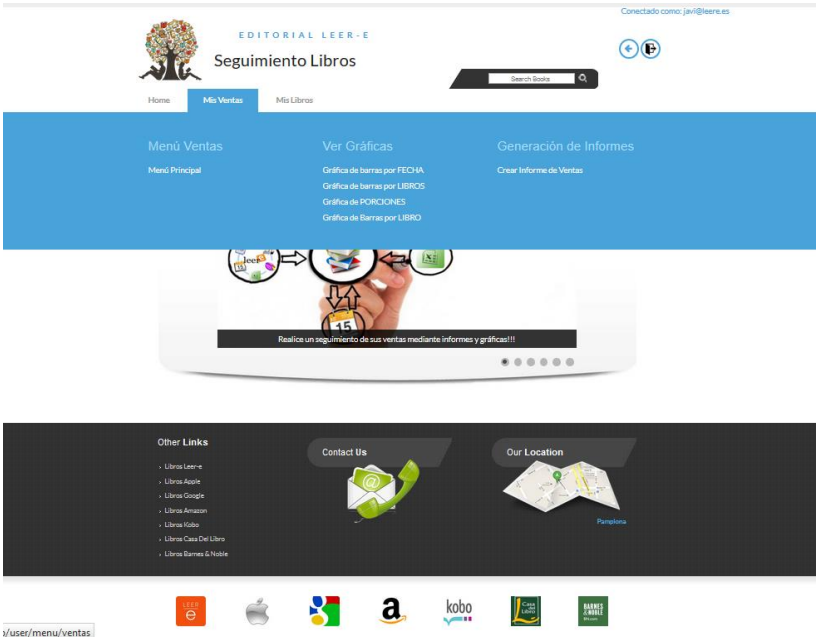
Login del Frontend:



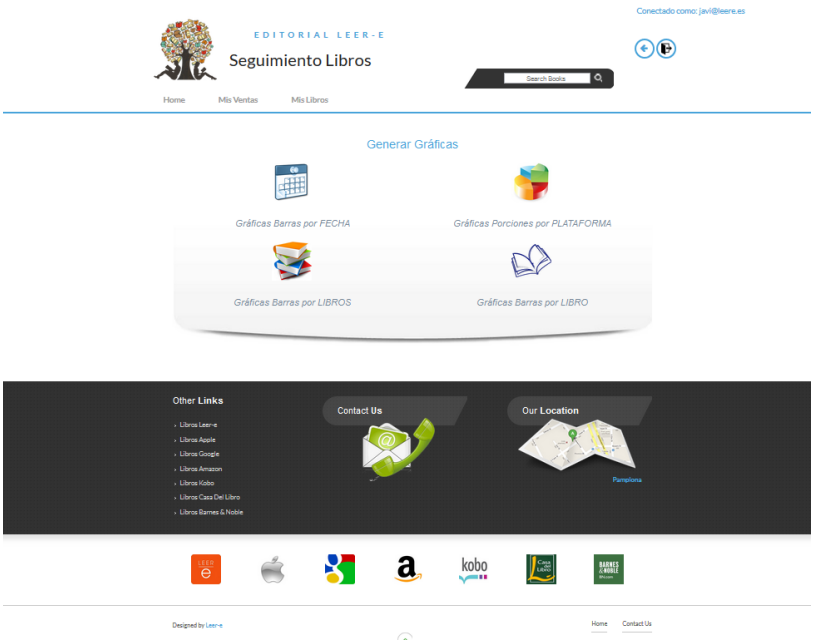
Menú principal del Frontend:



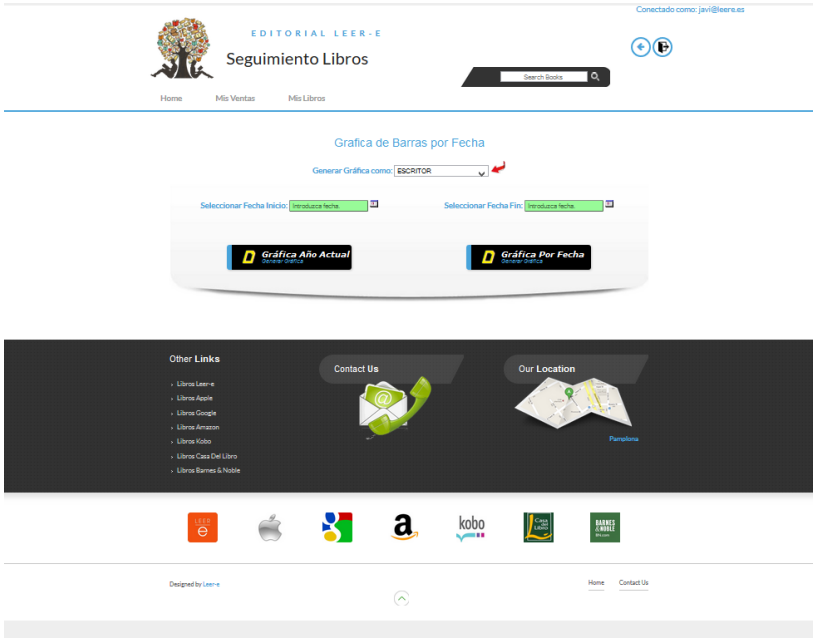
Menú desplegable Mis ventas:



Menú generar gráficas:



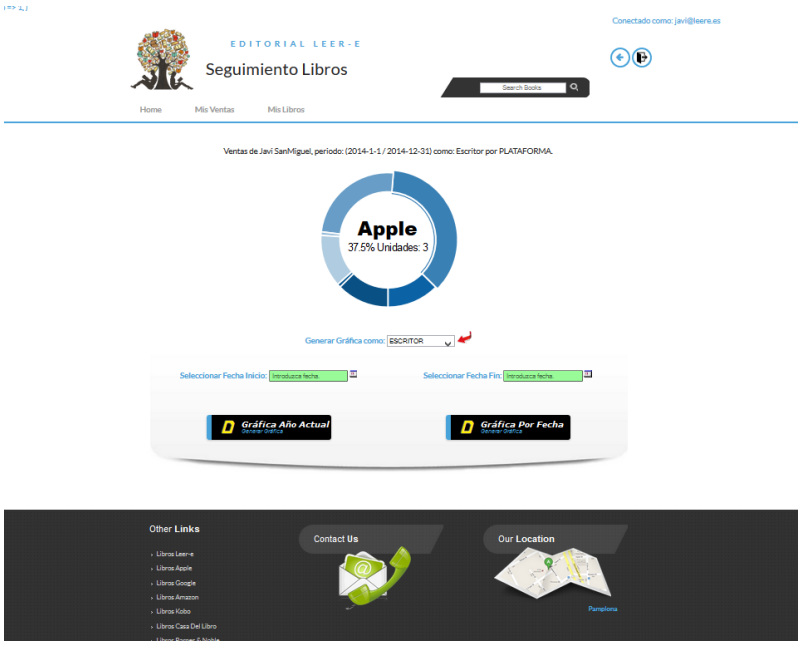
Menú filtrar gráficas:



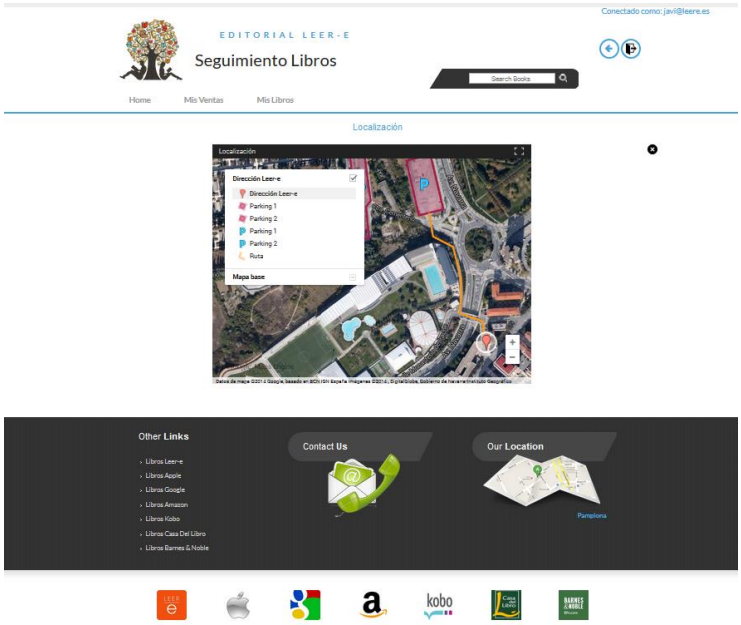
Gráfica de barras para libros:

Gráfica para ver ventas de un libro:

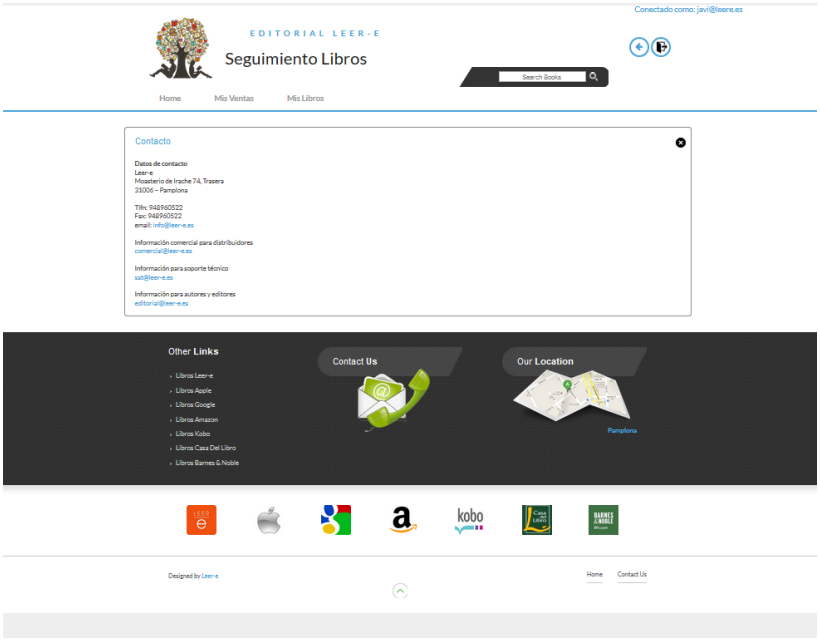
Gráfica porciones para ver ventas según plataforma:



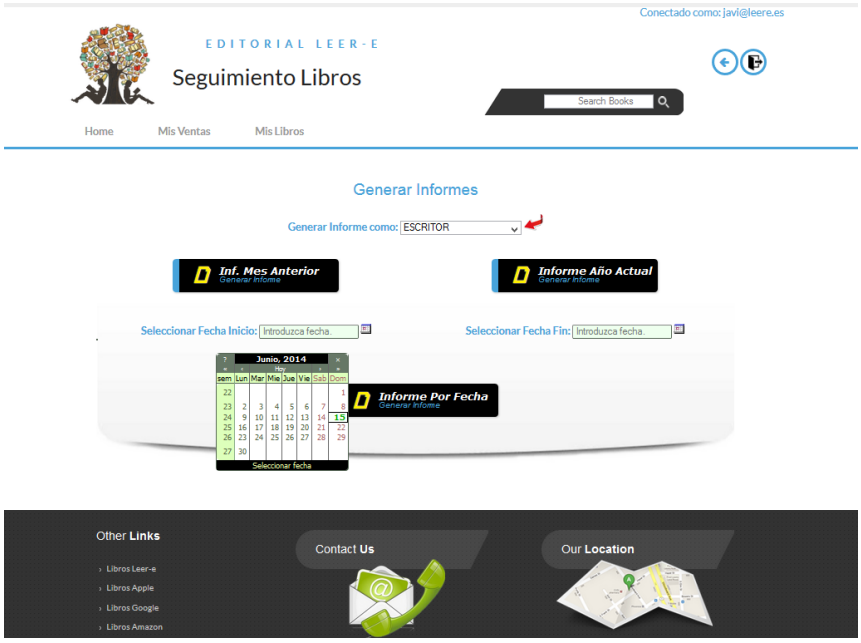
Localización de la empresa:



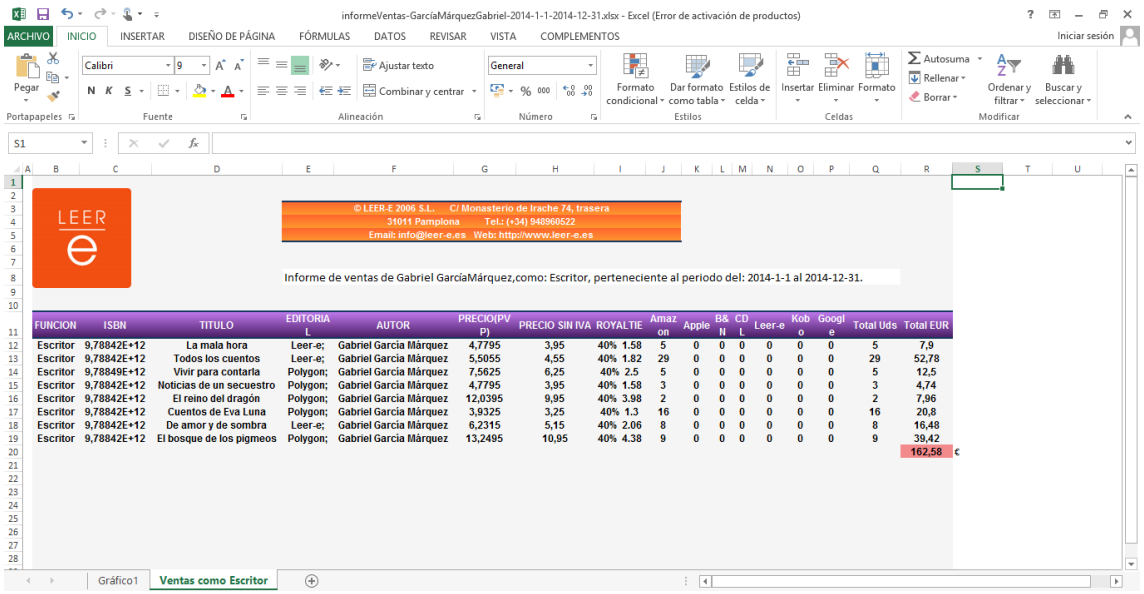
Datos de la empresa:



Vista para generar informe:



Ejemplo de informe:



FUNCION	ISBN	TITULO	EDITORIAL	AUTOR	PRECIO(PVP)	PRECIO SIN IVA	ROYALTIE	Amazon	Apple	B&N	CDL	Leer-e	Kobo	Google	Total Uds	Total EUR
Escritor	9,78842E+12	La mala hora	Leer-e;	Gabriel Garcia Márquez	4,7795	3,95	40%	1.58	5	0	0	0	0	0	5	7,9
Escritor	9,78842E+12	Todos los cuentos	Leer-e;	Gabriel Garcia Márquez	5,5055	4,55	40%	1.82	29	0	0	0	0	0	29	52,78
Escritor	9,78849E+12	Vivir para contarla	Polygon;	Gabriel Garcia Márquez	7,5625	6,25	40%	2.5	5	0	0	0	0	0	5	12,5
Escritor	9,78842E+12	Noticias de un secuestro	Polygon;	Gabriel Garcia Márquez	4,7795	3,95	40%	1.58	3	0	0	0	0	0	3	4,74
Escritor	9,78842E+12	El reino del dragón	Polygon;	Gabriel Garcia Márquez	12,0395	9,95	40%	3.98	2	0	0	0	0	0	2	7,96
Escritor	9,78842E+12	Cuentos de Eva Luna	Polygon;	Gabriel Garcia Márquez	3,9325	3,25	40%	1.3	16	0	0	0	0	0	16	20,8
Escritor	9,78842E+12	De amor y de sombra	Leer-e;	Gabriel Garcia Márquez	6,2315	5,15	40%	2.06	8	0	0	0	0	0	8	16,48
Escritor	9,78842E+12	El bosque de los pigmeos	Polygon;	Gabriel Garcia Márquez	13,2495	10,95	40%	4.38	9	0	0	0	0	0	9	39,42
																162,58 €