UNIVERSIDAD CENTRAL DE VENEZUELA

FACULTAD DE CIENCIAS

ESCUELA DE COMPUTACIÓN

Desarrollo de un Módulo de Sugerencias de Adquisición y Recomendación de Libros para la Aplicación Web de Gestión de la Bolsa del Libro de la Facultad de Ciencias de la UCV

Trabajo Especial de Grado presentado ante la
ilustre Universidad Central de Venezuela por el
Br. Edgar O. Vera F. para optar al
título de Licenciado en computación.

Tutor: Jaime Parada.

Caracas, Noviembre 2014

Quienes suscriben, miembros del jurado designado por el Consejo de la Escuela de Computación de la Facultad de Ciencias de la Universidad Central de Venezuela, para examinar el Trabajo Especial de Grado titulado: **Desarrollo de un Módulo de Sugerencias de Adquisición y Recomendación de Libros para la Aplicación Web de Gestión de la Bolsa del Libro de la Facultad de Ciencias de la UCV**, presentado por el bachiller Edgar O. Vera F., C.I.: 17.115.763, a los fines de cumplir con el requisito legal para optar por el título de Licenciado en Computación, dejan constancia de lo siguiente:

Leído el trabajo por cada uno de los miembros del jurado, se fijó el día Viernes 21 de Noviembre del 2014, a las 5:00 PM, para que su autor lo defendiera en forma pública, en la Sala 1 de la Escuela de Computación mediante una exposición oral de su contenido. Finalizada la defensa pública del Trabajo Especial de Grado, el jurado decidió aprobarlo con la nota de _____ puntos.

En fe de lo cual se levanta la presente Acta, en Caracas a los ventiún (21) días del mes de Noviembre del año dos mil catorce (2014).

Jaime Parada
Tutor

Prof. Marrero Carmen
Jurado Principal

Profa. Castillo Zenaida
Jurado Principal

RESUMEN

La Bolsa del Libro, es una dependencia adscrita a la Biblioteca Alonso Gamero, ubicada en la Facultad de Ciencias de la Universidad Central de Venezuela. Actualmente, lleva a cabo sus procesos de alquiler de libros, donaciones, registros de ejemplares, solicitud de solvencias y otros, a través de una aplicación Web que tiene dos años en producción.

En este Trabajo Especial de Grado se desarrollan modificaciones, actualizaciones y adiciciones de módulos al sistema Web de la Bolsa del Libro, solucionando las fallas detectadas previamente por los usuarios. Para ello, se automatizaron nuevos procesos en el módulo de estudiantes y el de administración. Adicionalmente, se implementó un módulo de profesores que permite almacenar sugerencias de libros que podrían ser adquiridos por la Bolsa del Libro.

Para llevar a cabo estos objetivos, se propone una adaptación del método ágil XP (Programación Extrema), el cual se caracteriza por su simplicidad y retroalimentación.

Palabras claves:

Bolsa del Libro, Aplicación Web.

Índice general

Introducción

La Bolsa del Libro de la Facultad de Ciencias, ha desarrollado una aplicación Web que es empleada para la gestión de procesos. Esta aplicación posee un módulo de estudiantes que les permite realizar el pre-alquiler de libros, solicitar solvencias, entre otras funcionalidades. Así mismo, existe un módulo de administración en el cual los encargados de la Bolsa pueden procesar automáticamente el alquiler asignando uno o varios ejemplares a los estudiantes, obtener un registro de los libros entregados, facilitar información de los libros que han sido devueltos, permitir el ingreso de nuevos ejemplares, donaciones, configuraciones, auditorías, así como obtener los reportes de alquileres, deudores y libros más solicitados, además de almacenar los usuarios que pueden acceder al sistema.

En el tiempo que lleva prestando servicios la aplicación de la Bolsa del Libro, sus usuarios han detectado fallas y la necesidad de implementar nuevos procesos que no habían sido contemplados. Por esta razón, se realiza este Trabajo Especial de Grado que tiene como objetivo principal actualizar el sistema de la Bolsa del Libro dando solución a problemas detectados, desarrollando nuevos procesos en los módulos de estudiantes, modificando las estadísticas del sistema en el módulo de administración, integrando nuevas reglas del negocio. De igual manera, se hacen mejoras en términos de eficiencia a los procesos de alguiler, reporte, es-

tablecimiento de jerarquías y provilegios, y ajustes de diseño. También se implementa un módulo de sugerencias, el cual le permitirá a los profesores proponer libros que puedan ser adquiridos por la Bolsa del Libro.

Estructura del Documento

A continuación se describe la estructura del documento:

- **Primera Parte: Planteamiento del Problema**

  Se plantea el problema con base en la investigación realizada y se define el objetivo, los objetivos específicos del trabajo a realizar y la justificación.

- **Segunda Parte: Marco Conceptual**

  Se describen los aspectos referentes a la Bolsa del Libro de la Facultad de Ciencias de la Universidad Central de Venezuela, desde su perfil, los servicios ofrecidos, los procesos que intervienen para llevar a cabo las tareas cotidianas. Además, se describe de forma más detallada desde la perspectiva de los estudiantes y los administradores el proceso de alquilar un libro como función primordial del sistema.

- **Tercera Parte: Marco Tecnológico**

  Se describen las tecnologías utilizadas, tales como el lenguaje de programación Ruby, Rails como el framework de desarrollo de aplicaciones Web, el sistema manejador de bases de datos MySQL, Git como sistema que utilizaremos para mantener las versiones de la aplicación, la metodología de desarrollo de software utilizada conocida como Programación Extrema (XP) y la herramienta de JasperReport que nos permitirá realizar los reportes del sistema.

- **Cuarta Parte: Marco Aplicativo**

  Se especifica la adaptación de las fases de XP y las actividades realizadas en cada una de las iteraciones que conforman el desarrollo del sistema, junto con las pruebas de aceptación las cuales fueron especificadas por el cliente.

- **Quinta Parte: Conclusiones y Recomendaciones**

  Se especifica los resultados obtenidos, objetivos alcanzados y recomendaciones para futuros trabajos.

Capítulo 1

Planteamiento del Problema

1.1. Descripción del Problema

El proceso manual que se desarrollaba anteriormente para la adquisición de libros, significaba un gran esfuerzo tanto para los estudiantes como para los administradores. Al sustituir este sistema por una aplicación Web para la gestion de procesos de la Bolsa del Libro [3], se automatizaron todos los procesos facilitando la prestacion de servicios, ahorrando tiempo y esfuerzo.

Los usuarios de la aplicación han presentado una serie de requerimientos y necesidades después de haber trabajado con el sistema desde su inicio, detectándose fallas y la ausencia de nuevas funcionalidades que vayan de la mano con el proceso de alquileres, con la solicitud de solvencias, presentación de reportes, almacenamiento de nuevos libros, ejemplares, y mostrar estadísticas que permitan el análisis del comportamiento del sistema. Adicionalmente se

busca una manera de interacción entre los profesores y el sistema de la Bolsa de Libros con la finalidad de obtener sugerencias para adquisición de nuevos libros.

1.2. Objetivo

Objetivo General

Actualizar la aplicación de gestión de la Bolsa del Libro e implementar el módulo que permita a los profesores realizar sugerencias.

Objetivos Específicos

- Analizar los procesos de la Bolsa del Libro que han presentado fallas.
- Implementar nuevas reglas de negocio en el módulo de administración.
- Desarrollar nuevas funcionalidades en la administración de alquileres, ejemplares, reportes y estadísticas.
- Diseñar e implementar un estructura de control de acceso basado en roles en el módulo de administración.
- Diseñar e implementar un módulo de sugerencias que cubra las necesidades de los profesores al momento de proponer a la Bolsa del Libro un libro para que sea adquirido.
- Implementar una funcionalidad que permita calificar una sugerencia.

1.3. Justificación

Hasta el momento se ha observado que es necesario implementar nuevas funcionalidades que estaban presentes en el proceso manual de la Bolsa del Libro y no se contemplaron en el desarrollo de la aplicación Web para la gestion de procesos, con la finalidad de obtener un sistema más completo, que permita ahorrar tiempo, dedicación y esfuerzo, agilizando todas las tareas administrativas, disminuyendo la carga laboral al personal administrativo, además de facilitar al estudiante el acceso a los servicios ofrecidos por la Bolsa del Libro. Por esta razón es necesario capturar los requerimientos que aparecieron desde que se esta usando la aplicacion Web, para posteriormente diseñar un módulo que permita realizar sugerencias para la adquisición de nuevos libros.

1.4. Alcance

Actualizar una Aplicación Web de la Bolsa del Libro de la Facultad de Ciencias de la Universidad Central de Venezuela que facilita la tarea de alquilar libros tanto para los estudiantes como para los administradores es nuestra prioridad, mantener los niveles de funcionamiento y mejorar procesos que se estan utilizando.

Esta actualización permitira:

- Solucionar problemas presentados en el proceso de alquiler durante el tiempo en servicio de la aplicación.
- Mejorar procesos del módulo de administración referentes al area de alquileres, ejemplares, estadísticas, reportes y otros.

- Implementar soluciones a los nuevos requerimientos presentados por el personal de la Bolsa del Libro.
- Desarrollar un nuevo módulo que cumpla las necesidades de los profesores con respecto a la Bolsa del Libro.
- Integrar como prioridad en el módulo de profesores un area de sugerencia de libros para ser adquiridos por por la Bolsa del Libro.

Capítulo 2

Marco Conceptual

En este capítulo se muestran las bases que fundamentan el desarrollo de este Trabajo Especial de Grado. Se comienza describiendo la funcionalidad de la Bolsa del Libro, detallando los procesos que la integran y que fueron automatizados en el sistema que se modifica y actualiza en este trabajo. Se explica detalladamente las operaciones que componen módulo de administración y el de estudiantes, mostrando el proceso de alquiler de libros, mediante diagramas de flujo para identificar todos los pasos y puntos de decisión que se realizan hasta completarse cada operación.

2.1. Antecedentes

La Bolsa del Libro es una dependencia adscrita a la Biblioteca Alonso Gamero de la Facultad de Ciencias cuyas actividades están enmarcadas en el alquiler semestral de libros de la Biblioteca.

En periodos anteriores han existido diferentes planes y proyectos de automatización referentes a la Bolsa, sin embargo no fue hasta el 2012 que se creó un sistema Web para el control de los procesos de La Bolsa del Libro que actualmente está en producción.

En el año 2007 M. Araujo y C. Pereira [1] desarrollaron una aplicación de comercio electrónico utilizando Herramientas de Software Libre, implementando un sistema de información vía Web y un sitio Web para la Bolsa del Libro de la Facultad de Ciencias, con el objetivo principal de automatizar los procesos de manipulación de información, responder a los requerimientos de administración de la Bolsa del Libro, automatizar el proceso del alquiler y compra de libros permitiendo realizar el pago en línea del servicio a los estudiantes de la Facultad de Ciencias de la Universidad Central de Venezuela.

Para el año 2011 A. Jiménez y J. Oviedo [2] diseñaron una aplicación de web para la automatización de los procesos de la Bolsa del Libro la cual constaba con módulos de alquiler, libros, ejemplares, pre-alquiler, solvencias, usuarios, auditorias, configuraciones y listados de inventario. Esta solución no llegó a implementarse.

En el año 2012 se desarrolló una nueva aplicación Web que buscaba satisfacer las necesidades de la Bolsa del Libro, que no habían sido solventadas en trabajos anteriores. Para ello se diseñó e implementó una aplicación de gestión de procesos y servicios de la Bolsa del Libro de forma automatizada [3]. El trabajo de investigación anterior a este se tomó en consideración para crear un nuevo sistema con una versión de Rails más actualizada, un mejor modelo de base de Datos y el uso de herramientas de generación de reportes más adecuadas. Este sistema esta ctualmente en producción y será el punto de partida para los aportes de este Trabajo Especial de Grado. A continuación se describen los módulos que compoten este sistema.

2.2. Módulo de Administradores

Los usuarios que tienen acceso a este módulo, son los encargados de ingresar toda la información al sistema, el ingreso de libros y sus ejemplares, configuraciones, solvencias, acceso de usuarios y más. Además, en esta área se culmina el proceso de alquiler que iniciaron los estudiantes, permitiéndoles obtener los libros solicitados y guardar un registro digital de dicha transacción.

Durante el curso del semestre en este módulo se tiene acceso a gran cantidad de información estadístico, que permite tener una idea de que libros son más solicitados, cantidad de usuarios que han realizado alquileres agrupados por escuela, datos que al final ayudarán en la toma de decisiones.

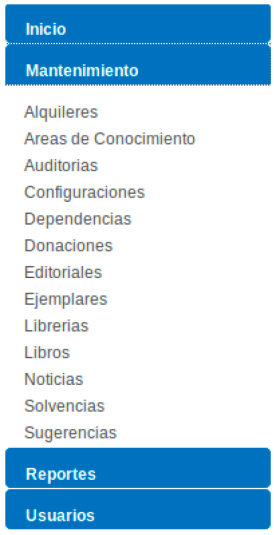


Figura 2.1: Panel derecho módulo de Administration

El panel derecho (ver Figura 2.1) contiene el menú de la aplicación, el cual se divide en:

- **Inicio**: Muestra al usuario la vista principal del administrador.
- **Mantenimiento**: Esta sección contiene una lista de opciones, las cuales permiten configurar y mantener actualizado el sistema.
- **Reportes**: dentro de esta sección se listan los reportes disponibles de la aplicacion.
- **Usuarios**: En esta sección se realizan todas las operaciones referentes a los usuarios del sistema.

El panel izquierdo solo se encuentra disponible en la vista principal del administrador.

En la Figura 2.2 se refleja el total recaudado durante el periodo actual de alquiler y un listado de los libros más alquilados durante dicho periodo.

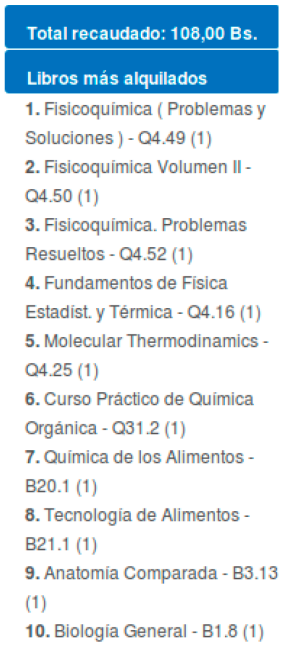


Figura 2.2: Panel Izquierdo Modulo de Administración

En la vista principal del módulo de administración, se pueden observar las estadísticas del sistema durante el proceso de alquileres, entre las cuales estan:

Libros alquilados por área de conocimiento: En la Figura 2.3 podemos observar la distribución de libros alquilados agrupados por área de conocimiento.

Figura 2.3: Libros alquilados por área de conocimiento

Ejemplares entregados a cada escuela: En la Figura 2.4 podemos observar la cantidad de libros entregados, agrupados por la escuela del estudiante que retira el libro.

Figura 2.4: Ejemplares entregados a cada escuela

Cantidad de usuarios por escuela: En la Figura 2.5 podemos visualizar la cantidad de usuarios que han realizado un alquiler agrupados por escuela.

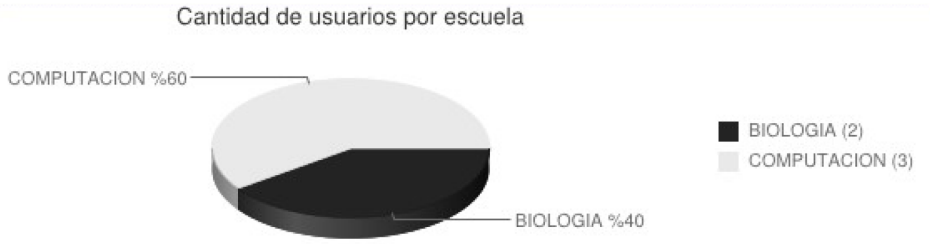


Figura 2.5: Cantidad de usuarios por escuela

El sistema posee una serie de parámetros que deben ser establecidos para que la aplicación funcione correctamente, esto se maneja desde el area de configuraciones, en la cual podran acceder a todas las configuraciones que se han establecido en el sistema hasta la fecha.

El menú principal de las configuraciones muestra un listado de las configuraciones que se han creado durante el uso de la aplicación, el cual posee las opciones para crear, editar, mostrar y eliminar. Al seleccionar la opcion de crear una nueva configuración, se despliega un formulario con los siguientes campos: Figura 2.6.

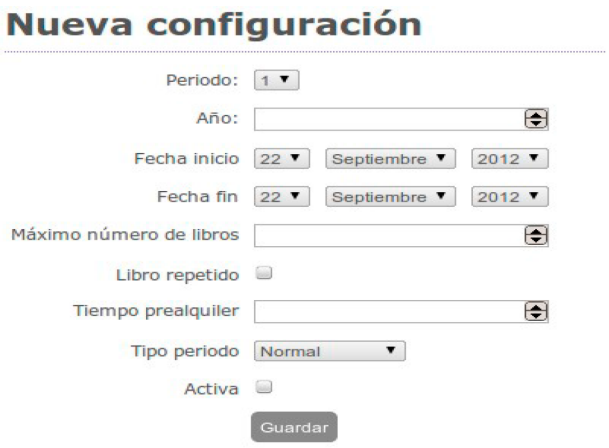


Figura 2.6: Nueva configuración

- **Período**: Se debe seleccionar el período al cual pertenece la configuracion, disponiendo de dos opciones (1 o 2).
- **Año**: Año durante la cual la configuración es válida.
- **Fecha de inicio**: Se debe seleccionar la fecha de inicio exacta desde cuando es válida la configuración.
- **Fecha fin**: Se debe seleccionar la fecha de culminación exacta hasta cuando es válida la configuración.
- **Máximo número de libros**: Indica el número máximo de libros que puede alquilar un estudiante durante el lapso de tiempo definido para la configuración.
- **Tiempo prealquiler**: Este campo debe contener el tiempo expresado en horas, durante el cual los libros estarán reservados, al momento de realizar un prealquiler.
- **Tipo período**: Este campo posee dos opciones disponibles, normal y extraordinario, siendo extraordinario los periodos intensivos y normal el resto.
- **Activa**: Al marcar este campo se indica que la configuración es la que está activa actualmente.

Las fechas no deben estar dentro de un lapso de tiempo que ya posea una configuración, solo debe existir una configuración activa y no se podrán realizar alquileres si no existe una configuración válida para la fecha en la cual se intente realizar el alquiler, estas son algunas de las restricciones para poder crear una configuracion correctamente.

En la sección de libros se crean, modifican y eliminan los títulos disponibles en la Bolsa del Libro. Los libros que sean agregados serán empleados para ingresar posteriormente los

ejemplares y estaran asociados.

El menú principal cuenta con un listado de los títulos agregados al sistema, el cual posee las opciones para crear, editar, mostrar y eliminar.

Al seleccionar la opcion de crear un libro, se permitira incluir un nuevo título dentro del sistema, por lo que es necesario haber agregado editoriales, áreas de conocimiento y dependencias anteriormente en el sistema.

En la Figura 2.7 se muestra un formulario de ejemplo con los siguientes campos:

Figura 2.7: Nuevo Libro

- **Buscar por ISBN**: Este campo será usado para buscar en "Google Books" los datos del libro a través de su ISBN.

- **Título**: Título del libro que se incluirá.
- **Autor**: Autor del libro que se incluirá.
- **Imagen**: Agrega una imagen almacenada en el computador, la cual será mostrada en los catálogos de libros.
- **Edición**: Numero de la edición que incluirá.
- **Lugar de impresión**: Lugar de impresión del libro.
- **Año**: Año del libro que será incluido.
- **ISBN**: Numero Internacional Normalizado del Libro.
- **Editorial**: Seleccione una de las editoriales cargadas en el sistema.
- **Área de conocimiento**: Seleccione una de las áreas de conocimiento cargadas en el sistema. A través de este campo se podrán filtrar los catálogos, distribuyendo los libros por su área de conocimiento.
- **Dependencia**: Seleccione una de las dependencia cargada en el sistema.

Entre las restricciones existentes para crear un libro correctamente estan:

- La cota y el ISBN deben ser únicos. La cota, título, edición y año son campos obligatorios.

En la sección de ejemplares se crean y modifican los ejemplares disponibles en la Bolsa del Libro, cada uno representa un libro físico dentro de la Bolsa del Libro.

El menú principal cuenta con un listado de los ejemplares agregados al sistema, el cual posee las opciones para crear, editar, mostrar y eliminar.

Al seleccionar la opción de crear, ésta permite incluir un nuevo ejemplar dentro del sistema.

Figura 2.8: Nuevo Ejemplar

En la Figura 2.8 se muestra un formulario de ejemplo. Los ejemplares deben estar asociados a títulos de libros previamente cargados en el sistema y los campos a rellenar son:

- **Libro**: Se debe indicar el libro al cual estará asociado el ejemplar.

- **Número de ejemplar**: Identificador único para el ejemplar asociado a un libro.
- **Costo de Alquiler**: Costo en bolívares del alquiler del libro por un semestre.
- **Tipo de adquisición**: Forma mediante la cual el ejemplar fue adquirido por la Bolsa del Libro.
- **Fecha de Ingreso**: Fecha en la cual ingreso el ejemplar a la Bolsa del Libro.
- **Observación**: En caso de existir
- **Nro. de Factura**: En caso de existir, se indica el número de factura del ejemplar.
- **Estatus Ejemplar**: Se indica el estatus que tendrá el ejemplar.
- **Librería**: Librería de la cual proviene el ejemplar.

En la seccion de solvencias el administrador del sistema podrá emitir las solvencias solicitadas por los estudiantes, disponiendo para ello de un listado donde se encuentran todas las solicitudes. El listado del cual dispondrá el administrador del sistema se puede observar en la Figura 2.9. Los administradores tiene la opción de marcar una solvencia como entregada, cuando se ha hecho.

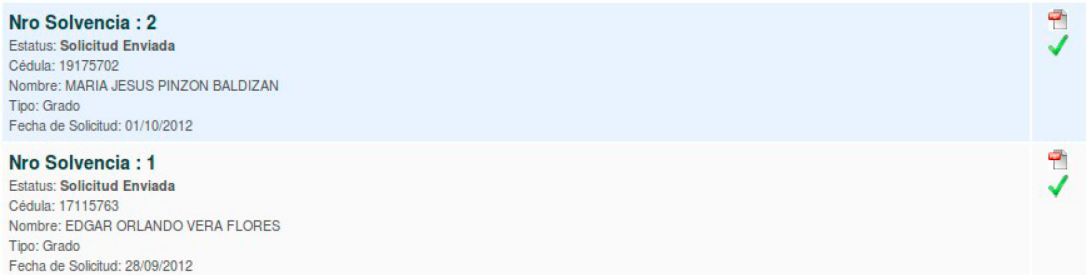


Figura 2.9: Lista de Solvencias

El sistema cuenta con una serie de reportes que permiten ver información relevante de las operaciones hechas en el sistema. Los reportes con los que cuenta el sistema son los siguientes:

- **Alquileres por escuela**: Muestra el número de total alquileres por escuela.
- **Ejemplares alquilados**: emite un listado de todos los ejemplares que se encuentran alquilados agrupados por escuela.
- **Libros adquiridos**: muestra los libros adquiridos en un intervalo de tiempo establecido por el usuario.
- **Libros alquilados**: muestra el total de libros alquilados, la cantidad y agrupados por escuela.
- **Libros sin alquilar**: emite un listado de los libros que no han sido alquilados en un lapso de dos años.
- **Listado de deudores**: listado con los deudores de la Bolsa del Libro.
- **Libros disponibles**: listado de todos los libros disponibles, agrupados por escuela.

Los usuarios existentes pueden ser modificados y eliminados desde el area de usuarios, además de ser creados si es necesario.

La opción que permite crear un nuevo usuario a través de un formulario se muestra en la Figura 2.10, los campos son los siguientes:

Figura 2.10: Nuevo Usuario

- **Nombre**: Se debe indicar el nombre del usuario.
- **Cédula**: Identificador único del usuario.
- **Clave**: Combinación de caracteres que permite autenticarte en el sistema.
- **Correo**: Dirección de correo electrónico del usuario.
- **Unidades de crédito**: Número que indica cuantas unidades de creditos ha aprobado el usuario.
- **Rol**: Debe indicar funcionalidad que tiene este usuario en el sistema. Puede ser administrador, estudiante, profesor o inactivo.
- **Dependencia**: Se indica la escuela a la cual pertenece el usuario.

Las restricciones para crear un usuario correctamente son las sigueintes:

- Cedula, nombre, clave, correo y rol son campos obligatorios.

- Cedula y correo deben ser únicos.
- El usuario no debe tener alquileres o solicitudes cargadas en el sistema.

2.3. Módulo de Estudiantes

Los usuarios autenticados en este módulo tendrán la opción de visualizar todos los libros disponibles en la bolsa del libro y además podrán pre-alquilarlos, para luego retirarlos físicamente en la Bolsa del Libro.

Entre los otros servicios que los estudiantes pueden acceder es al área de solicitud de solvencias, cambio de clave, alquileres realizados y solvencias solicitadas. Todas estas áreas la verán detallas en las siguientes secciones.

En la vista principal de este módulo se puede visualizar un catálogo de libros donde se muestran todos los libros disponibles. El catálogo muestra la cantidad existente, titulo, autor, edición, año y costo. Además cuenta con la posibilidad de agrupar por área de conocimiento como se puede ver en la Figura 2.11.

Existe la opción de agregar, que permite al estudiante seleccionar un libro y este ser almacenado para ser alquilado. Los libros seleccionados se mostrarán en el panel superior izquierdo de la página.

Las restricciones al momento de agregar un libro son:

- No se pude agregar dos veces el mismo libro.

- No puede exceder el número máximo de libros estipulados por la administración de la Bolsa del Libro.

Figura 2.11: Catalogo de Libros

Existe un panel superior izquierdo que muestra los libros que han sido seleccionados del catálogo de libros, mostrando su título y costo asociado, como se observa en la Figura 2.12, solo aparecerá si al menos se ha seleccionado un libro para agregar. Ademas, podrán realizar las siguientes acciones:

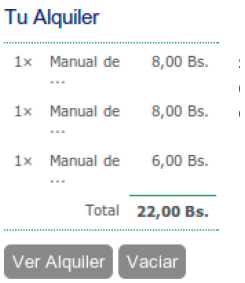


Figura 2.12: Carrito de Libros

- **Vaciar**: Esta acción elimina todos los libros agregados al alquiler en curso.
- **Ver alquiler**: Esta opción permite ver la información completa de los libros seleccionados, así como eliminar los libros que deseemos suprimir de nuestro alquiler.

Al seleccionar "alquilar" se muestra el listado de libros que tienes en tu carrito, como se observa en la Figura 2.13 y entre las acciones que se pueden realizar estan:

Figura 2.13: Resumen de alquiler

- **Eliminar**: Esta acción desvincula el libro de su carrito de libros y hace disponible el mismo para otros usuarios.
- **Alquilar**: Esta acción guarda el alquiler en curso, colocándole un estatus de "Prealquilado", cuando el usuario retire los libros de la Bolsa del Libro, entregando el recibo de pago, el estatus del alquiler cambiara a "Alquilado" y una vez que entregue todos los libros su estatus cambiará a "Alquiler Finalizado".

  Al confirmar el alquiler podremos ver un mensaje que indica que el alquiler se ha realizado exitosamente y un enlace para observar el comprobante, como se observa en la Figura 2.14.

UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
BIBLIOTECA ALONSO GAMERO
BOLSA DEL LIBRO

COMPROBANTE DE ALQUILER
PERIODO: 1 - 2012

DATOS DEL ALQUILER

NRO. ALQUILER:	339
FECHA DE EMISIÓN:	23/10/2012
FECHA DE ALQUILER:	23/10/2012

APELLIDOS Y NOMBRES: MARIA JESUS PINZON BALDIZAN
CEDULA: 19175702

COTA	TITULO	AUTOR	COSTO DE ALQUILER
Q8.7.2	Manual de Mineralogía	Dana - Hurlbut	60,00Bs.
Q8.8.1	Manual de Mineralogía.	Klein /Hurlbut. (Dana)	80,00Bs.
E1.7.1	Macroeconomía	Diulio, Eugene	50,00Bs.

PRECIO TOTAL: 190. Bs.

Figura 2.14: Comprobante de Alquiler

La Figura 2.15 contiene el menú de los diferentes módulos que posee el usuario con un rol de estudiante y muestra las siguientes opciones:

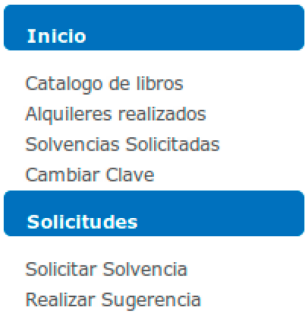


Figura 2.15: Panel Derecho de Estudiantes

- **Catálogo de libros**: Permite visualizar todos los libros disponibles en la Bolsa del Libro.

- **Alquileres realizados**: Muestra una lista con todos los alquileres realizados por el usuario, como se puede observar en la Figura 2.16.

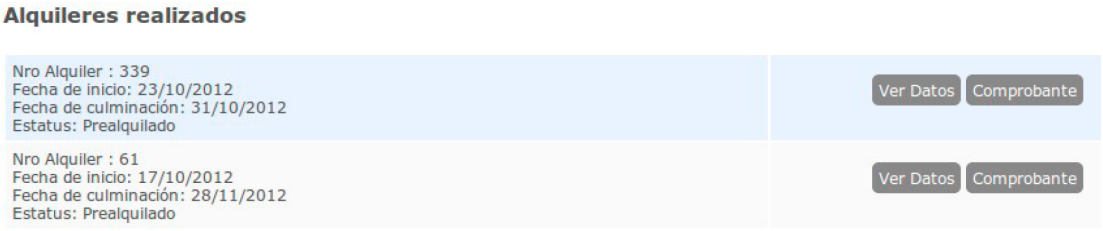


Figura 2.16: Alquileres Realizados

- **Solvencias solicitadas**: Muestra un listado con todas las solicitudes de solvencia hechas por el usuario, como se puede observar en la Figura 2.17.

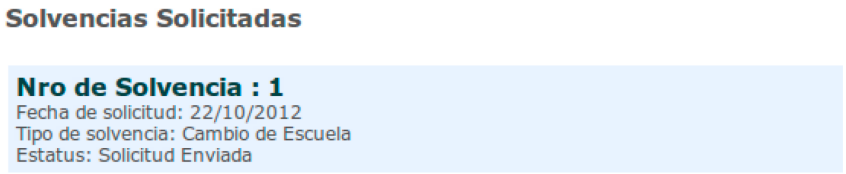


Figura 2.17: Solvencias Solicitadas

- **Solicitar solvencia**: Los usuarios cuentan con la posibilidad de realizar peticiones de solvencia con la Bolsa del Libro, para ello pueden hacerlo a través del siguiente formulario, donde especificarán el motivo de la petición. Ver Figura 2.18.

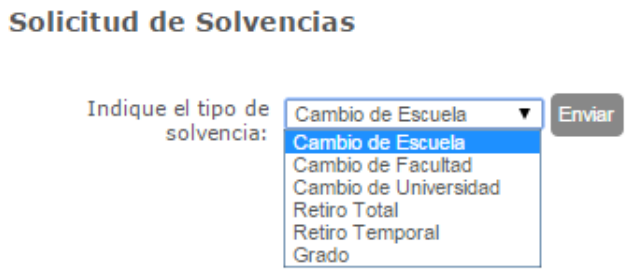


Figura 2.18: Solicitud de Solvencias

2.4. Alquileres (Nivel Funcional - Modulo de Estudiantes)

A partir de aquí, se mostrará como un usuario observa las diferentes operaciones que puede realizar a través del módulo del estudiante y el conjunto de pasos que se realizan por detrás de la interfaz al ejecutar cada operación.

Cada operación se ilustrará a través de un diagrama de flujo, que muestra desde el inicio (cuando se selecciona una opción a traves de la interfaz), punto de tomas de decisiones (puntos donde pueden ocurrir flujos alternos) hasta el final, que es donde se indica al usuario si la ejecución de la operación fue satisfactoria o no.

En la Figura 2.19 se engloba todas las operaciones que pueden realizar los usuarios durante el proceso de alquilar libros y mas adelante se detallará el comportamiento de cada uno de estos.

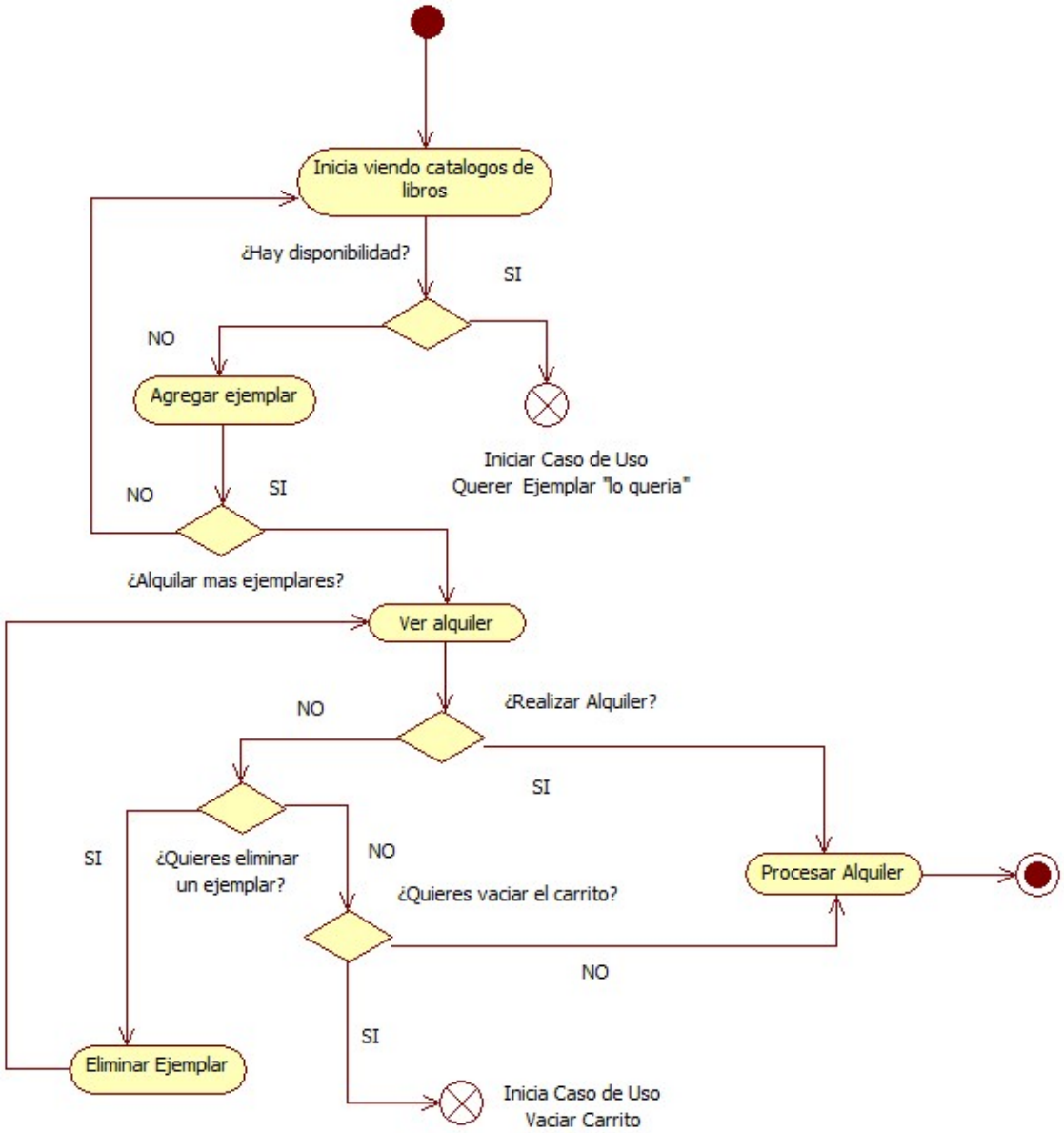


Figura 2.19: Flujo de actividades del proceso de alquiler - Modulo del Estudiante

Desde que inicia el proceso visualizando el catálogo de libros, el usuario puede pasar por diferentes pasos donde puede ejecutar funcionalidades como agregar un ejemplar, ver alquileres, procesar alquileres, vaciar carrito, eliminar un ejemplar o simplemente hacer destacar que un libro que está agotado él "lo quería". Se realizarán las operaciones siempre y cuando las condiciones estén dadas como lo muestra el diagrama de flujo.

Figura 2.20: Catalogo de libros. Carrito de Libros

En la Figura 2.20 se observan las 4 opciones que tienen los usuarios: **agregar ejemplar**, **lo queria**, **ver alquiler** y **vaciar carrito**, cada una desencadena una serie de acciones como veremos a continuación.

El comportamiento del sistema cuando un estudiante selecciona “agregar un libro” se puede observar en la Figura 2.21, donde se reflejan las condiciones que pueden generar flujos alternos hasta completar el proceso donde se le informa al usuario si se agregó o no el libro a su carrito.

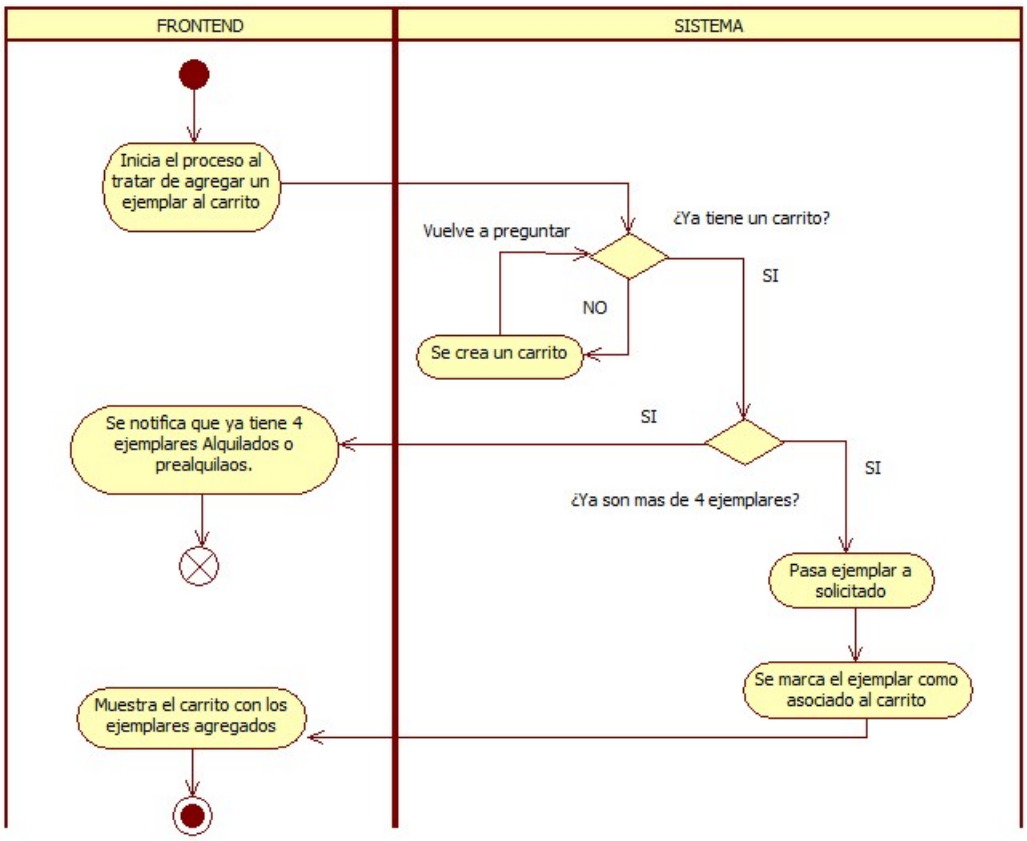


Figura 2.21: Agregar Ejemplar al Carrito

En la Figura 2.22 se refleja la situación en la que el estudiante selecciona el botón “Lo quería” y se inicia el proceso de almacenar la cantidad de veces que algún usuario requere este libro al no estar disponible.

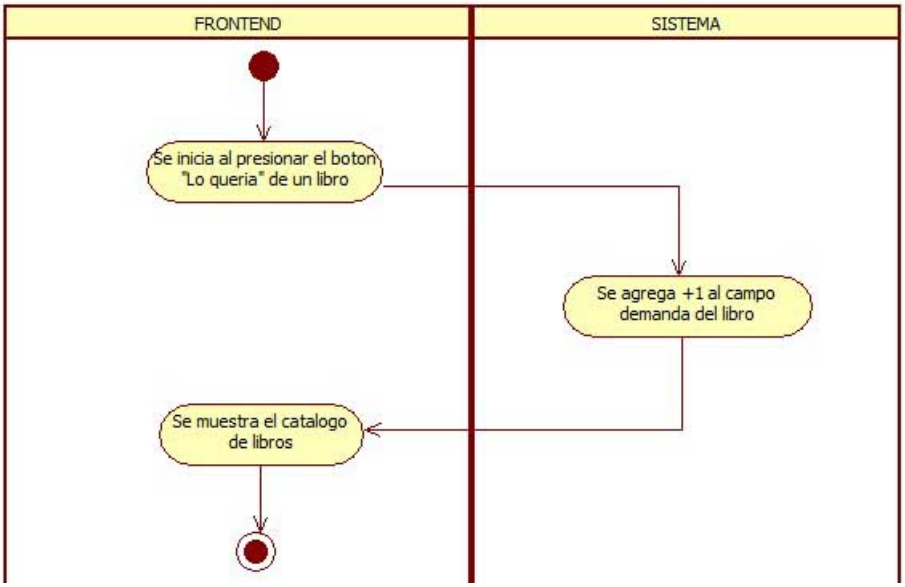


Figura 2.22: Contador de solicitudes de libro después de agotado

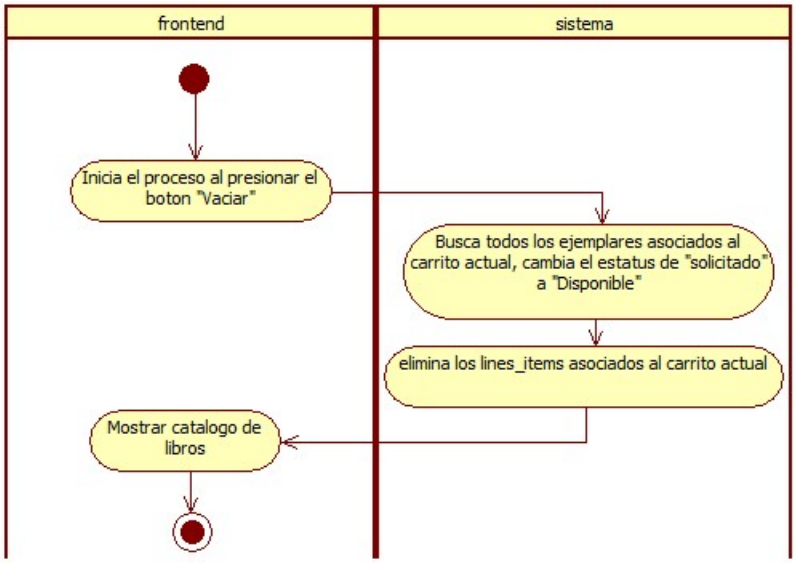


Figura 2.23: Vaciar carrito

Si el estudiante selecciono "vaciar" el carrito, esto inicia el proceso donde todos los ejemplares relacionados con el carrito (alquiler) pierden esta relación y cambiar su estatus a "Disponibles", como podemos observar en la Figura 2.23.

Al seleccionar el botón "Ver alquiler" se despliega la vista de la Figura 2.24, que refleja todos los ejemplares agregados al carrito y las 2 únicas operaciones que puede realizar: alquilar o eliminar ejemplares.

Figura 2.24: Vista del usuario ejemplares agregados al carrito

En la Figura 2.25, se muestra el comportamiento del sistema cuando el estudiante decide seleccionar "Alquilar", donde los ejemplares pasan a un estatus "pre-alquilado" y se crea por cada ejemplar un registro en la tabla devoluciones con estado "sin devolver".

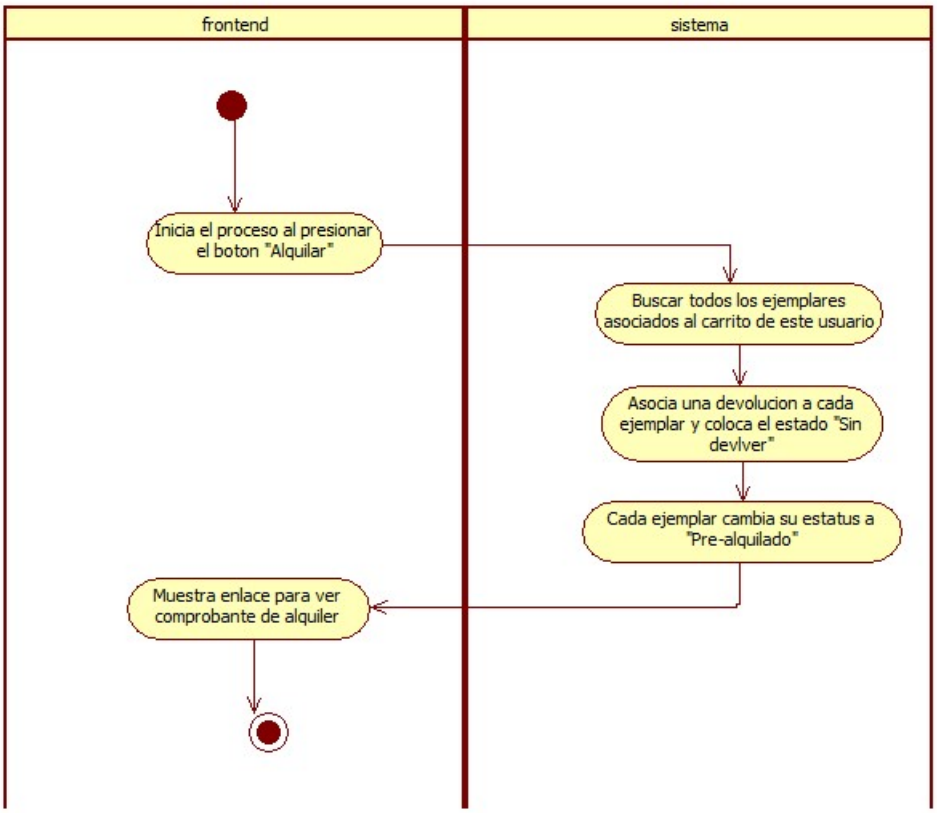


Figura 2.25: Ejecucion de Alquiler.

Cuando se toma la decisión de eliminar un ejemplar del carrito, desaparece la relación ejemplar-carrito y este cambia su estatus a “disponible” para que pueda ser alquilado por otro usuario, como se muestra en la Figura 2.26.

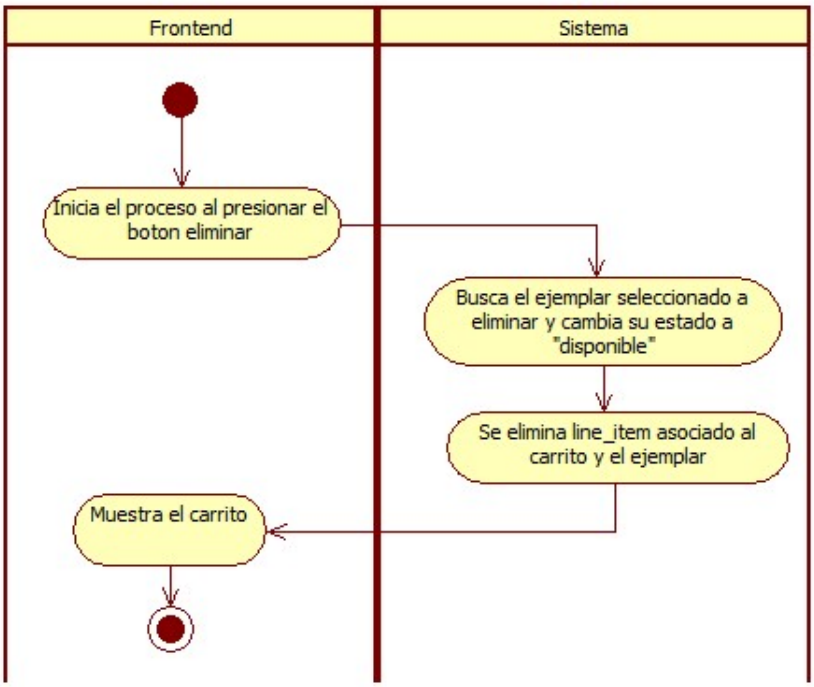


Figura 2.26: Eliminar un ejemplar seleccionado

2.5. Alquileres (Nivel Funcional - Modulo del Administrador)

En esta sección se mostrará como un usuario observa las diferentes operaciones que puede realizar a través del módulo de administración y el conjunto de pasos que se realizan por detrás de la interfaz al ejecutar cada operación. Toda la información generada por los estudiantes al realizar los pre-alquileres se encuentra en este módulo, de forma tal que los administradores por medio del sistema puedan entregar los libros y si ya se encuentran los

libros alquilados, procesar la devolución de los mismos. Este flujo se refleja en la Figura 2.27.

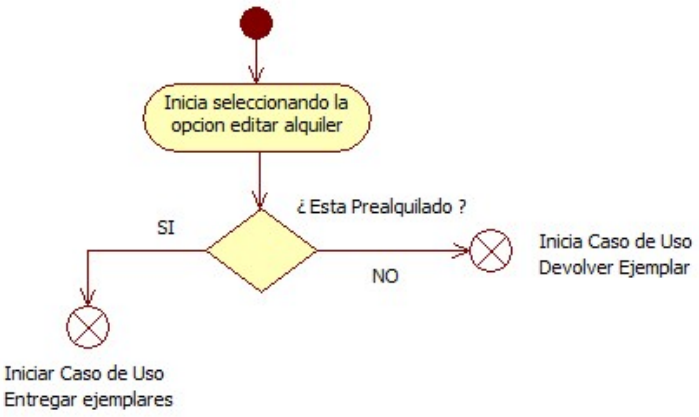


Figura 2.27: Proceso Post Alquiler (Administradores)

En la Figura 2.28 se observa como el administrador detalla los libros que un estudiante pre-alquilo y ahora se dispone a entregar el ejemplar fisicamente. Una vez el estudiante haya recibido los libros y este conforme, el administrador registra esta acción en el sistema a través del botón "Entregar libros".

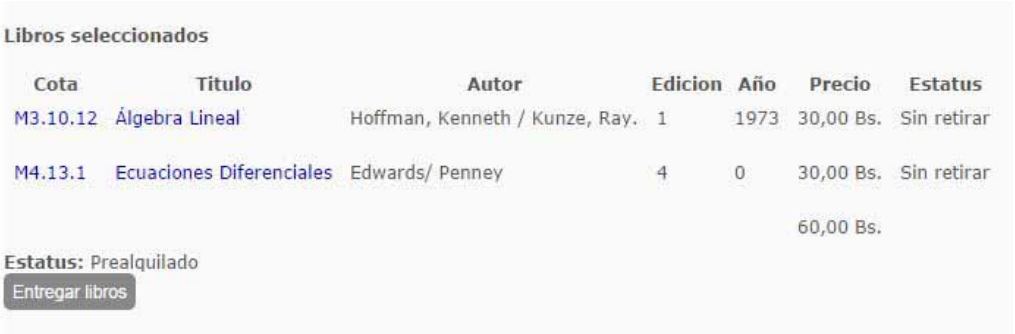


Figura 2.28: Funcionalidades Administrativas

Si el usuario decide seleccionar “Entregar libros” todos los ejemplares que estén relacionados con este alquiler pasan al estatus “Alquilado” y el alquiler cambia a estado “Alquilado”, como se puede ver en el flujo de la Figura 2.29.

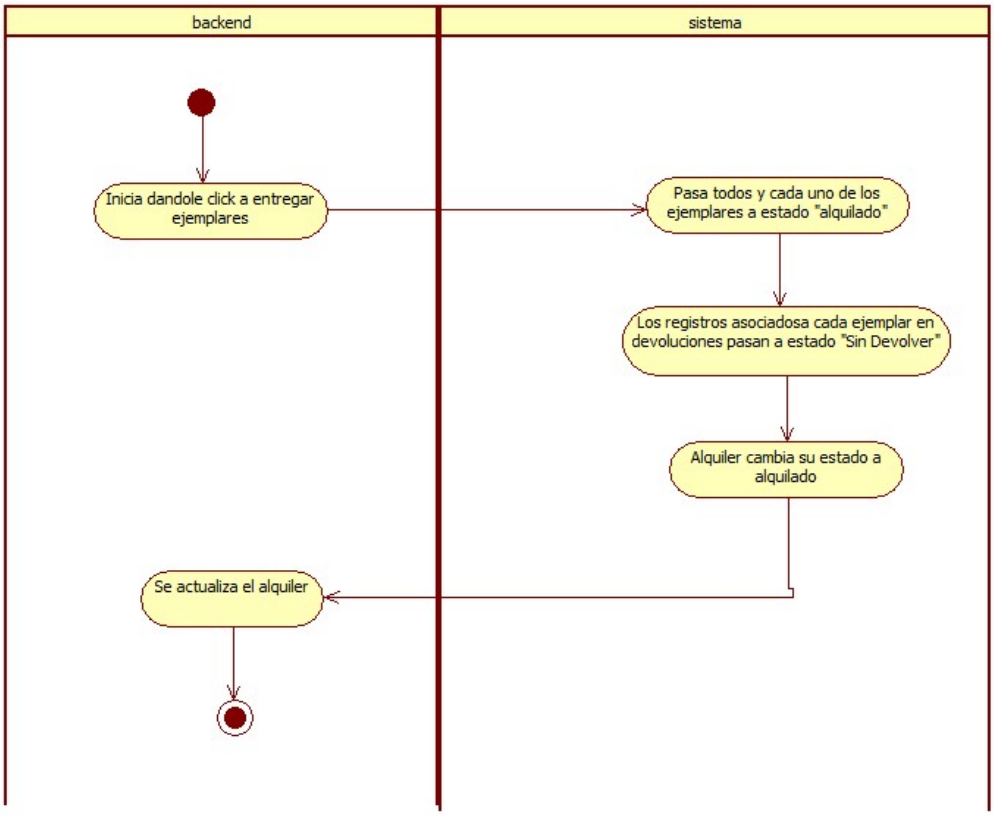


Figura 2.29: Entrega de Alquileres

Cuando un estudiante se dirige a la Bolsa del Libro con la intención de devolver un libro, el administrador busca el alquiler relacionado y al editarlo, la vista que se desplegará se observa en la Figura 2.30, en la cual se muestra la única opción que tiene el usuario, “devolver” libros.

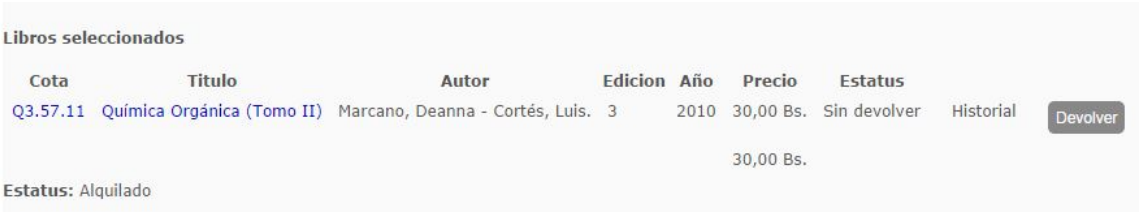


Figura 2.30: Funciones para editar alquiler

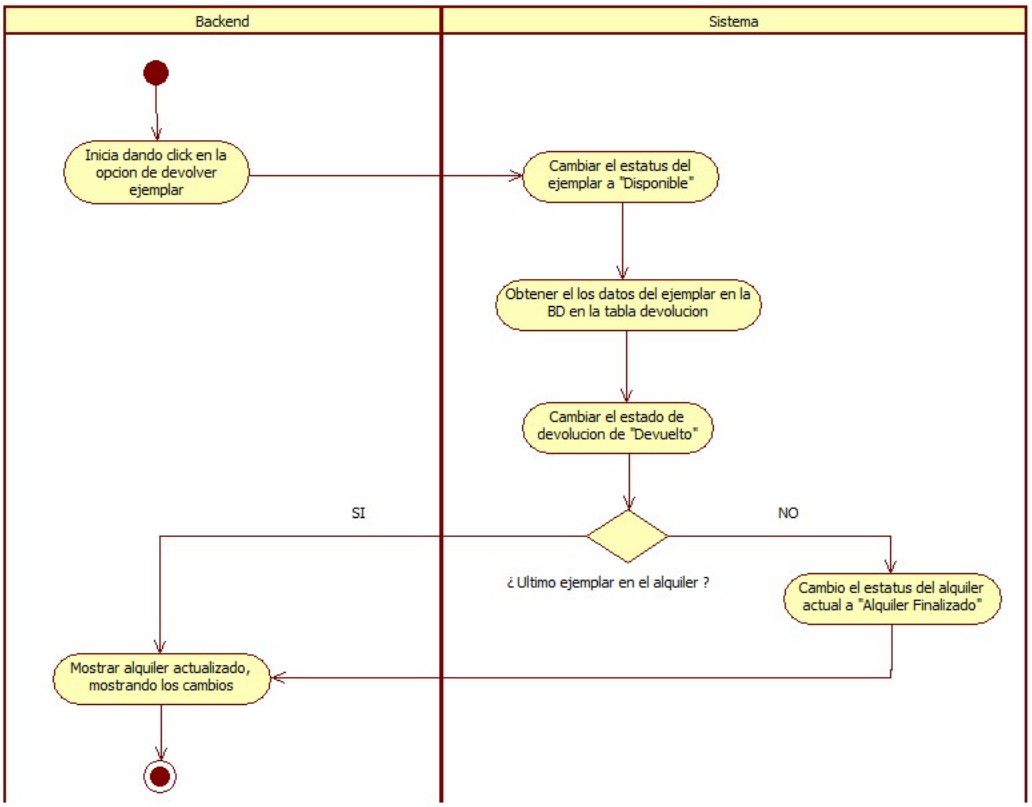


Figura 2.31: Proceso Devolver Ejemplares

Cuando el administrador selecciona el botón “devolver” el sistema inicia los cambios de registros de las tablas ejemplares y devoluciones que sean necesarios para completar el proceso y se considera el caso en el cual el comportamiento pueda variar realizando un paso extra que modifique la tabla alquileres, como se muestra en la Figura 2.31.

Capítulo 3

Marco Tecnológico

En este capítulo se describen las herramientas tecnológicas utilizadas en el desarrollo de la aplicación Web, tales como: Ruby, Rails, MySQL, Git, Jquery y JasperReport, además se hace una breve descripción de la metodología de desarrollo de software usada (Programación Extrema).

3.1. Ruby

Es un lenguaje de scripts, multiplataforma, totalmente orientado a objetos. Ruby es un lenguaje de programación interpretado y su implementación oficial es distribuida bajo una licencia de software libre.

Ruby es considerado un lenguaje flexible, ya que permite a sus usuarios alterarlo libremente. Las partes esenciales de Ruby pueden ser quitadas o redefinidas a placer. Se puede agregar funcionalidad a partes ya existentes. Ruby intenta no restringir al desarrollador.

Ruby tiene un conjunto de funcionalidades entre las que se encuentran las siguientes: [4]

- Posibilidad de redefinir los operadores (sobrecarga de operadores).
- Manejo de excepciones, como Java y Python, para facilitar el manejo de errores.
- Escribir extensiones en C para Ruby es más fácil que hacer lo mismo para Perl o Python, con una API muy elegante para utilizar Ruby desde C. Esto incluye llamadas para embeber Ruby en otros programas, y así usarlo como lenguaje de scripting.
- Ruby no necesita declaraciones de variables. Se utilizan convenciones simples para nombrar y determinar el alcance de las mismas. Posee cuatro niveles de ámbito de variable: global, clase, instancia y local.
- Puede cargar bibliotecas de extensión dinámicamente si lo permite el sistema operativo.
- Tiene manejo de hilos (threading) independiente del sistema operativo. De esta forma, tienes soporte multi-hilo en todas las plataformas en las que corre Ruby, sin importar si el sistema operativo lo soporta o no.
- Ruby es fácilmente portable: se desarrolla mayoritariamente en GNU/Linux, pero corre en varios tipos de UNIX, Mac OS X, Windows 95/98/Me/NT/2000/XP, DOS, BeOS, OS/2, etc.

3.2. Ruby on Rails (RoR)

Es un entorno de desarrollo web de código abierto que está optimizado para la satisfacción de los programadores y para la productividad sostenible. Permite escribir un buen código

evitando que se repita trabajo y favoreciendo la convención antes que la configuración.

El desarrollo sobre este entorno está basado en dos filosofías, no te repitas (del inglés Don't repeat yourself, DRY) y convención sobre configuración. DRY significa que las definiciones deberían hacerse una sola vez. Dado que Ruby on Rails es un framework de pila completa, los componentes están integrados de manera que no hace falta establecer puentes entre ellos. Mientras que convención sobre configuración significa que el programador sólo necesita definir aquella configuración que no es convencional. [8]

A continuación se explica brevemente cada uno de los componentes del patrón de diseño arquitectónico MVC.

Modelo

Representa los datos de la aplicación y las reglas para manipular esos datos, en caso de Rails se utilizan principalmente para la gestión de las tablas de base de datos que en la mayoría de los casos corresponde a un modelo en la aplicación, contiene la lógica del negocio.

Vista

Es la lógica de visualización, es decir, cómo se muestran los datos provenientes del Controlador. Con frecuencia en las aplicaciones Web la vista consiste en una cantidad mínima de código de algún lenguaje incluido en HTML. El método que se emplea en Rails por defecto es usar Ruby Embebido (archivos .rhtml, desde la versión 2.x en adelante de RoR archivos .html.erb), que son básicamente fragmentos de código HTML código en Ruby.

Controlador

Administra la navegación entre vistas, procesa las peticiones del usuario desde el navega-

dor web e invocan los modelos de datos mostrando los resultados de dichas peticiones en una plantilla (template) o un redireccionamiento. La implementación del Controlador es manejada por el ActionPack de Rails, que contiene la clase ApplicationController. Un controlador en Rails debe heredar de esta clase y definir las acciones como métodos de dicha clase.

Los componentes de Rails

Action Pack es una gema que contiene Action Controller, Action View y Action Dispatch. La parte "VC" de "MVC".

Rails está conformado por componentes individuales para el desarrollo web, entre estos: [9]

- **Action Controller**: Maneja los controladores de la aplicación, procesando las peticiones, extrae los parámetros y los envía a la acción prevista. Incluye la administración de sesiones y la gestión de redireccionamiento.
- **Action View**: Maneja las vistas de la aplicación, puede ser creada con HTML y XML, soporta Ajax y Css. Implementa la función de los layouts, que permite especificar un grupo de elementos que se mostrarán en todas las páginas.
- **Action Dispatch** : gestiona el enrutamiento de las solicitudes web y las despacha como desee, ya sea a su aplicación o cualquier otra aplicación Rack.
- **Active Record**: Maneja los modelos de datos, proporciona independencia de la Base de Datos, funcionalidades básicas CRUD (Create, Read, Update and Delete o en español, Crear, Leer, Actualizar y Eliminar), capacidad de búsqueda, relaciona los modelos entre sí, entre otros servicios, etc.

- **Action Mailer**: Es un framework utilizado para dar servicio de correo electrónico, para enviar, recibir y procesar los correos electrónicos, sin formato o integrando partes sobre plantillas flexibles.
- **Active Resource**: Es un framework para la gestión de la conexión entre los objetos de negocio y servicios web. Se implementa una forma de mapa de recursos basados en objetos locales con la semántica CRUD.
- **Railties**: Es el código del núcleo Rails que crea nuevas aplicaciones y los conecta con los framework en una sola aplicación.
- **Active Support**: Es una extensa colección de clases y extensiones de la biblioteca estándar de Ruby que se utilizan en Rails, tanto por el código del núcleo como de sus aplicaciones.

3.3. MySQL

MySQL es un sistema manejador de bases de datos relacional (SMBDR), el cual proporciona un servidor de base de datos SQL (Structured Query Language), multihilo, multiusuario y robusto.

MySQL es capaz de almacenar enormes cantidades de datos, de gran variedad y de distribuirlos para cubrir las necesidades de cualquier tipo de organización, desde pequeños establecimientos comerciales a grandes empresas y organismos administrativos.

MySQL es capaz de soportar que varios usuarios se conecten al mismo tiempo y que puedan manipular y administrar las distintas bases de datos existentes y administrar al

SMBDR como tal, en el mismo instante de tiempo.

MySQL implementa un protocolo que se encarga de emplear diferentes algoritmos para cifrar los datos que viajan a través de una red pública (ejemplo: Internet), el cual tiene por nombre Capa de Seguridad de Sockets (Secure Sockets Layer - SSL). Este protocolo trabajaría en el cliente y el servidor MySQL. Así mismo, maneja gestión de usuarios y contraseñas, en el sentido de qué usuarios tienen acceso a qué tablas y con qué permisos, manteniendo de esta forma un buen nivel de seguridad de los datos.

Los clientes pueden conectar con el servidor MySQL usando Sockets TCP/IP en cualquier plataforma. La interfaz para el conector ODBC (My-ODBC) proporciona a MySQL soporte para programas clientes que usen conexiones ODBC (Open Database Connectivity).

Proporciona sistemas de almacenamiento transaccionales y no transaccionales (no permiten hacer "ROLLBACK"). MySQL en su versión 5.0 tiene como característica el manejo de integridad referencial y de transacciones, gracias al motor InnoDB, que permite el manejo de transacciones mediante la sentencia "BEGIN WORK" y finaliza con un "COMMIT" o "ROLLBACK", o puede terminar en modo "AUTOCOMMIT". Para trabajar con este motor es necesario especificarlo al momento de la creación de cada una de las tablas, declarando explícitamente que van a ser del tipo InnoDB.

El sistema de seguridad en MySQL es referido como el Sistema de Privilegios de Acceso [10]. Permite la autenticación de los usuarios del servidor de MySQL y la verificación de las actividades de todos los usuarios sobre el servidor y las bases de datos. La seguridad en MySQL es aplicada en dos niveles: Nivel de servidor y Nivel de Base de Datos. Cuando un usuario trata de acceder a una base de datos, primero se verifica si el usuario tiene privilegio para acceder al servidor de base de datos, después el servidor verifica si el usuario tiene

privilegios para conectarse a una base de datos. La verificación de conexión al servidor y la verificación de conexión a la base de datos son dos procesos que MySQL siempre lleva a cabo.

MySQL realiza la verificación de privilegios del servidor y la base de datos usando unas tablas del sistema llamadas tablas de concesión. Estas tablas contienen toda la información necesaria para aplicar las políticas de seguridad convenientes. Todos los host (otros computadores) y usuarios que se conectan al servidor MySQL deben estar representados en las tablas de concesión.

3.4. Git

Git es un sistema de control de versiones o Version Control System (VCS) libre, de código abierto, distribuido y diseñado para manejar desde pequeños a grandes proyectos con rápidez y eficiencia.

Características de GIT

- La función de Git que realmente lo hace destacar, es su modelo de ramificación; permitiendo tener varias ramas locales que pueden ser completamente independientes el uno del otro.
- Git es rápido. Con Git, casi todas las operaciones se realizan a nivel local, lo que supone una ventaja enorme en la velocidad en sistemas centralizados que constantemente tienen que comunicarse con un servidor en algún lugar.
- Git se distribuye. Esto significa que en lugar de hacer un “checkout” de la cabecera

actual del código fuente, hace un "clon" de todo el repositorio. El modelo de datos que usa Git garantiza la integridad criptográfica de cada parte de su proyecto.

- Git tiene la "zona de espera" o "índice". Esta es una zona intermedia en la que se compromete que puede ser formateado y revisado antes de terminar el compromiso. Git es liberado bajo la licencia de código abierto GPLv2.

3.5. JasperReports

JasperReports es una herramienta creada por la compañía JasperSoft, diseñada para la creación de reportes que permite entregar contenido enriquecido al monitor, impresora o archivos PDF, XML, HTML, XLS Y CSV. Está escrito completamente en Java y puede ser usado en aplicaciones Web para generar contenido dinámico. Su propósito principal es ayudar a crear páginas orientadas, documentos listo para imprimir de una manera simple y flexible.

JasperReports, así como muchas aplicaciones de reportes, usa plantillas estructuradas en múltiples secciones, tal como título, resumen, detalle y página y grupos de encabezado y pies de página. Cada sección tiene un diseño de forma libre en donde puedes colocar varios tipos de elementos, incluyendo imágenes, campos de texto dinámicos y estáticos, líneas y rectángulos. El motor de reportes usa esta plantilla para organizar datos en un archivo XML (JRXML) o crearlo usando un API de librerías. Estos datos pueden venir de distintos orígenes de datos, incluyendo bases de datos relacionales, colecciones, o arreglos de objetos Java o datos XML (ver Figura 3.36). Los usuarios se pueden conectar a la librería de reportes en una fuentes de datos hechas a la medida implementando una simple interfaz. [7]

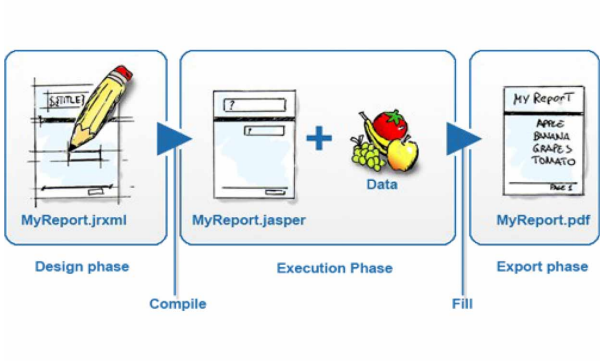


Figura 3.36: Proceso de compilación de un reporte

Ireport

Ireport es un programa para el diseño de reportes visuales que hace uso de JasperReports. Esta librería es un motor de reportes que puede ser integrada a una aplicación libre o comercial para generar los reportes diseñados con Ireport, mostrarlos en la pantalla o exportarlos en un formato final como PDF, OpenOffice, DOCX y muchos otros. Alternativamente, se puede transmitir el resultado a través de una aplicación Web o mandar el documento final directamente a una impresora. JasperReport de alguna manera viene siendo el núcleo de Ireport. [5]. Ireport permite diseñar reportes, JasperReports ejecutar los reportes y generar una salida en una aplicación de Java.

3.6. Jquery

jQuery es una biblioteca de JavaScript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas

web.

jQuery es software libre y de código abierto, posee un doble licenciamiento bajo la Licencia MIT y la Licencia Pública General de GNU v2, permitiendo su uso en proyectos libres y privados. jQuery, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio.

En la Figura 3.37 se muestra un ejemplo de como se puede lograr un mismo efecto usando Javascript simple y JQuery.

JavaScript

```
var ejemplo = null; // objecto
function mover() {
  ejemplo.style.left = parseInt(
      ejemplo.style.left)+1+'px';
  setTimeout(mover,20); //
      llamado a mover en 20ms
}
function init() {
  //obteniendo el objeto 'ejemplo
      '
  ejemplo = document.
      getElementById('ejemplo');
  //asignando la posicion inicial
       en 0px
  ejemplo.style.left = '0px';
        //empezando animacin
  mover();
}
window.onload = init;
```

JQuery

```
1 $("#element").animate({ left: 100
    px; });
```

Figura 3.37: Animación en Javascript simple y JQuery

3.7. Programación Extrema (XP)

La programación extrema o eXtreme Programming (de ahora en adelante, XP) es una metodología de desarrollo de la ingeniería de software formulada por Kent Beck. Es el más destacado de los procesos ágiles de desarrollo de software. Al igual que éstos, la programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. Los defensores de la XP consideran que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos. Creen que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos.

Se puede considerar la programación extrema como la adopción de las mejores metodologías de desarrollo de acuerdo a lo que se pretende llevar a cabo con el proyecto, y aplicarlo de manera dinámica durante el ciclo de vida del software.

Practicas

Las doce prácticas de la programación extrema tienen su origen en prácticas bien conocidas en la ingeniería del software:

1. **El juego de la Planificación**: Poner en producción las características más importantes.

2. **Versiones Pequeñas**: periódicamente, se producen nuevas versiones del sistema.
3. **Metáfora del Sistema**: descripción general del sistema.
4. **Diseño Simple**: el sistema se diseña con la máxima simplicidad posible.
5. **Pruebas Continuas**: los clientes especifican pruebas funcionales.
6. **Refactorización**: modificar la forma del código sin cambiar su funcionamiento.
7. **Programación por parejas**: la programación se realizan de a dos programadores por computadora.
8. **Posesión Colectiva del Código**: cualquier programador puede cambiar cualquier parte del sistema en cualquier momento.
9. **Integración continua**: los cambios se integran en el código base varias veces por día.
10. **Semana laboral de 40 horas**: cada trabajador trabaja no más de 40 horas por semana.
11. **Cliente en el Sitio**: el equipo de desarrollo tiene acceso todo el tiempo al cliente.
12. **Estándares de Codificación**: todo el código debe estar escrito de acuerdo a un estándar de codificación.

Valores

La programación extrema se apoya en cuatro valores fundamentales:

- **Comunicación**: La comunicación se realiza de diferentes formas. Para los programadores el código comunica mejor cuanto más simple sea. Si el código es complejo hay que esforzarse para hacerlo inteligible. El código autodocumentado es más fiable que los comentarios ya que éstos últimos pronto quedan desfasados con el código a medida que es modificado. Debe comentarse sólo aquello que no va a variar, por ejemplo el objetivo de una clase o la funcionalidad de un método.

- **Simplicidad**: La simplicidad es la base de la programación extrema. Se simplifica el diseño para agilizar el desarrollo y facilitar el mantenimiento. Un diseño complejo del código junto a sucesivas modificaciones por parte de diferentes desarrolladores hacen que la complejidad aumente exponencialmente. Para mantener la simplicidad es necesaria la refactorización del código, ésta es la manera de mantener el código simple a medida que crece.

- **Retroalimentación**: Al estar el cliente integrado en el proyecto, su opinión sobre el estado del proyecto se conoce en tiempo real. Al realizarse ciclos muy cortos tras los cuales se muestran resultados, se minimiza el tener que rehacer partes que no cumplen con los requisitos y ayuda a los programadores a centrarse en lo que es más importante.

- **Coraje**: Muchas de las prácticas implican coraje o valentía. Una de ellas es siempre diseñar y programar para hoy y no para mañana. Esto es un esfuerzo para evitar empantanarse en el diseño y requerir demasiado tiempo y trabajo para implementar el resto del proyecto. La valentía le permite a los desarrolladores que se sientan cómodos con reconstruir su código cuando sea necesario. Esto significa revisar el sistema existente y modificarlo si con ello los cambios futuros se implementaran más fácilmente.

- **Estándares de Codificación**: todo el código debe estar escrito de acuerdo a un estándar de codificación.

Actividades

A continuación se describen las actividades de XP: [12]

- **Planificación**: Se definen entre el conjunto de desarrolladores y usuarios del sistema una serie de Historias de Usuario que describen las funcionalidades requeridas para el software que se construirá. El desarrollo se divide en iteraciones, cada iteración comienza con un plan para que se elijan las Historias de Usuario a desarrollar.
- **Diseño**: Los diseños deben ser sencillos, si alguna parte del sistema es de desarrollo complejo, lo apropiado es dividirla en varias. Si hay fallos en el diseño o malos diseños, estos deben ser corregidos cuanto antes.
- **Codificación**: XP argumenta que lo más importante para el desarrollo del producto de software es codificar. Codificar en XP es hacer diagramas que generaran código, scripts para una Aplicación Web o código para una Aplicación que necesita compilarse.
- **Pruebas**: Hay que asegurarse de que todo lo que se hace funcione correctamente. Para ello, lo mejor es desarrollar las pruebas desde el momento que se conocen las Historias del Usuario.

3.8. Adaptación XP

A continuación se describen las actividades que se realizarán para la adaptación del proceso XP que se utilizó durante el desarrollo de la aplicación Web.

3.8.1. Iteraciones

El método XP propone dividir el trabajo en iteraciones, las cuales se enfocan en versiones parciales del sistema hasta llegar al producto final. Los nuevos requerimientos son recibidos progresivamente y son incluidos en una nueva iteración. Las iteraciones pueden ser de dos tipos principalmente: por objetivos o por lapsos de tiempo. En el desarrollo de este Trabajo Especial de Grado, las iteraciones están basadas en objetivos. Los objetivos planteados en cada iteración se establecen según su prioridad.

3.8.2. Planificación

Según XP, la actividad de planificación comienza creando una serie de Historias de Usuario, en las cuales se describen en una o dos oraciones los requerimientos del sistema en terminología del cliente, proporcionando a su vez una estimación del tiempo necesario para el desarrollo. [12]

En este trabajo, el formato para escribir las Historias de Usuario se presenta en el Cuadro 1. La primera columna corresponde al número que sirve como identificador, la segunda es la fecha de inicio y la tercera una breve descripción de los requerimientos que se trabajarán.

ID	FECHA	DESCRIPCION

Cuadro 3.1: Lista de historias

3.8.3. Diseño

El método de desarrollo Programación Extrema, propone la fase de diseño como una guía de implementación para una historia de usuario determinada.

Para esta fase, XP recomienda la creación de prototipos y/o diagramas, cuando establecer un diseño para una historia de usuario resulte complicado. Asimismo, promueve aplicar refactorización, que permita realizar mejoras al diseño del código después que se ha escrito. [12]

3.8.4. Codificación

Durante la actividad de codificación, este método sugiere la programación en pareja, que consiste en dos personas en una misma estación de trabajo desarrollando el código de una historia de usuario. Esto ayuda a seguir los estándares de programación, lo cual es otro aspecto requerido por el método de Programación Extrema. Por último se deben realizar frecuentes integraciones de código entre los grupos de trabajo, permitiendo evitar problemas de compatibilidad e interfaz y ayudando al descubrimiento de errores en el sistema. [12]

Para el desarrollo de esta aplicación, no se adopta la programación en pareja. Durante el desarrollo se mantiene la consistencia y legibilidad del código para facilitar la comprensión en el desarrollo de la aplicación, esto pensando en que dicho prototipo será mejorado y ampliado por otros desarrolladores. También se realiza una integración constante del código por medio del sistema de control de versiones Git.

En la presente adaptación del proceso de desarrollo se extraerán y mostrarán partes del código fuente que sean esenciales en la comprensión de la aplicación y la solución a los requerimientos de las historias de usuario.

3.8.5. Pruebas

Las pruebas serán de aceptación en donde el usuario o cliente pone a prueba el sistema y verifica que hayan quedado cubiertos todos los requerimientos.

Capítulo 4

Marco Aplicativo

Iteracion 0: Levantamiento de Información

En esta primera iteración se describe el resultado de las dos primeras reuniones con el equipo de la Bolsa del Libro, donde se recolectaron todas las peticiones de mejoras y cambios a realizar.

En una primera reunión se revisó el sistema y junto al listado de fallas se fueron señalando cómo está funcionando y cómo debería funcionar la aplicación. Después de esta reunión se inició el estudio del código de cada proceso que había presentado algún tipo de problema, se hizo un análisis de la base de datos y se procedió a establecer un ambiente de desarrollo similar al que se encuentra en producción.

En una segunda reunión, ya teniendo más experiencia de cómo funcionan ciertos procesos

del sistema se realizó una estimación de tiempo y prioridades de los requerimientos. Se plantearon posibles soluciones a ciertos casos y se solicitó más información sobre algunas reglas del negocio que se necesitarón a la hora de actualizar el sistema.

4.1. Iteracion 1: Carga de usuarios a la BD y cálculo de estadísticas

4.1.1. Planificación

Se muestran las historias de usuario de esta iteración.

ID	FECHA	DESCRIPCIÓN
1	10-04-2013	Se carga la lista de Control de Estudios a la BD y actualización de tabla usuarios.
2	10-04-2013	Reprogramar el cálculo de las estadísticas que se muestran en el módulo de administración.
3	10-04-2013	Permitir observar estadísticas y el contador de monto recaudado durante y después de finalizar el semestre.
4	10-04-2013	Por período se requiere mantener el valor de todas las estadísticas que se muestran al inicio del módulo de administración.

Cuadro 4.2: Historia de Usuario. Iteración 1.

4.1.2. Diseño

Carga de usuarios a la BD

La primera tarea que se completó en esta iteración fue la carga de nuevos usuarios en el sistema. Por ser la segunda vez que se realizaba (la primera fue cuando se puso en producción el sistema) requería mucha atención y pautas de cómo manejar los usuarios nuevos, antiguos y los que no están inscritos en este semestre.

Los usuarios no inscritos se colocaran como "Inactivo". Los usuarios que están en la tabla "usuarios"de la BD y en la nueva lista de Control de Estudios no se modificaron y aquellos que solo se encuentran en la lista de Control de Estudio se agregaron al sistema.

Actualización de estadísticas

Una vez se tiene la captura de datos correcta para conocer las estadísticas de los libros alquilados agrupados por área de conocimiento, ejemplares entregados a cada escuela y la cantidad de usuarios por escuela, solo hacía falta agregar un formulario donde se muestre los periodos anteriores, donde por defecto el periodo era el actual o el último cursado. Como observamos en la Figura 4.39.

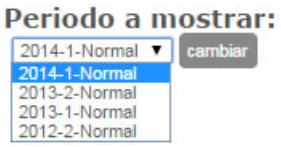


Figura 4.39: Selección de periodo para mostrar estadísticas

4.1.3. Codificación

La lista de usuarios viene dada por Control de Estudios en un archivo de formato Excel, el cual tiene todos los campos que se necesitan en la tabla usuarios. Lo primero que se hizo fue crear una tabla con la misma estructura de la tabla usuarios. Véase la Figura 4.40.

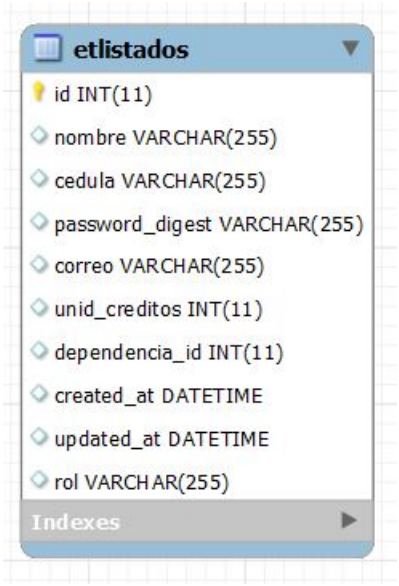


Figura 4.40: Estructura tabla de estudiantes inscritos en Control de Estudios

Una vez creada la nueva tabla se procedió a cargar la lista proporcionada por Control de Estudio en la tabla denominada "Etlistado". Esto se logró transformando el archivo a lenguaje SQL para insertar los datos en la BD. La función que cumple esta tabla es almacenar datos temporales, sin modificar directamente la tabla usuarios.

Posteriormente se ejecutaron una serie de consultas SQL que se encargaron de actualizar la tabla usuarios.

Lista de Usuarios Inactivos

En la Figura 4.41, se observa la consulta SQL que indicará los usuarios que pasaron a estado "inactivo" del sistema porque ya se graduaron, cambiaron de facultad o por alguna razón no están en la lista de estudiantes inscritos en este semestre.

```
DROP TABLE IF EXISTS inact_users;
CREATE TABLE inact_users AS SELECT *
FROM (SELECT u.cedula AS last_list,e.cedula AS new_list
FROM usuarios u
LEFT JOIN etilistados e ON u.cedula = e.cedula) view
WHERE new_list IS NULL;
```

Figura 4.41: Lista de Usuarios Inactivos

Lista de Usuarios Nuevos

En la Figura 4.42, se observa la consulta SQL que indica los usuarios que se agregaron como nuevos al sistema, que no se encontraban registrados en la Bolsa del Libro.

```
DROP TABLE IF EXISTS  new_users;
CREATE TABLE new_users AS SELECT *
FROM (SELECT u.cedula AS last_list,e.cedula AS new_list
FROM usuarios u
RIGHT JOIN etilistados e ON u.cedula = e.cedula) view
WHERE last_list IS NULL;
```

Figura 4.42: Lista de Usuarios Nuevos

Actualización de Usuarios

En la Figura 4.43, se observa la consulta SQL que actualizó la tabla de usuarios, pasando a estado "inactivo" aquellos que estaban en la tabla "inac_users".

```
UPDATE usuarios u, inact_users i SET rol = CASE WHEN rol = 'estudiante' THEN 'Inactivo' ELSE u.rol
END WHERE u.cedula = i.last_list;
```

Figura 4.43: Actualización de Usuarios: inactivos

Nuevos Usuarios de la Bolsa del Libro

En la Figura 4.44, se observa la consulta SQL que agregó a la tabla usuarios aquellos registros que se encontraban en la tabla "new_list".

```
INSERT INTO usuarios (`nombre`, `cedula`, `password_digest`, `correo`, `unid_creditos`,
`dependencia_id`, `created_at`, `updated_at`, `rol`, view)
SELECT e.`nombre`, e.`cedula`, e.`password_digest`, e.`correo`, e.`unid_creditos`,
e.`dependencia_id`, e.`created_at`, e.`updated_at`, e.`rol` , e.`view` FROM new_users n
JOIN etilistados e ON n.new_list = e.cedula;
```

Figura 4.44: Nuevos Usuarios de la Bolsa del Libro

Las condiciones que se establecieron para mostrar las estadísticas parecían correctas solo al inicio del período de alquiler, donde no había casi libros sin devolver. Las consultas estaban relacionadas a la tabla alquileres y devoluciones, cuando se entregaban los libros se perdían los datos, ya que el registro de devoluciones con estado "sin devolver" pasaba a "Devuelto".

Para solucionar el problema del cálculo del monto recaudado y los gráficos estadísticos del sistema, se tomaron los resultados solo aquellos alquileres que se encontraban condicionados al estado "Alquilado" o "Alquiler finalizado". Este resultado se agrupo por periodo usando el campo çonfiguracion_idz obtenemos la información necesaria para mostrar las estadisticas correctamente.

En las Figuras 4.45, 4.46, 4.47 y 4.48 se observan las consultas SQL que se realizaron usando la lógica descrita anteriormente para mostrar las estadísticas (gráficos porcentuales) y el monto recaudado.

```
###Libros alquilados agrupados por area de conocimiento
  @ejemplares_area = Ejemplar.find_by_sql(["select count(e.id) as cantidad, b.area_conocimiento_id, c.nombre
    from alquileres a join lines_items l on (a.id = l.alquiler_id)
    join devoluciones d on (l.id = d.line_item_id)
    join ejemplares e on (l.ejemplar_id = e.id)
    join libros b on (e.libro_id = b.id)
    join areas_conocimientos c on (b.area_conocimiento_id = c.id)
    where (a.estatus = 'Alquilado' or a.estatus = 'Alquiler Finalizado') and a.configuracion_id = ?

    group by b.area_conocimiento_id", params[:id_periodo]])
```

Figura 4.45: Libros alquilados agrupados por área de conocimiento

```
#Libros alquilados agrupados por escuela de alumno solicitante

        @ejemplares_escuela = Ejemplar.find_by_sql(["select count(e.id) as cantidad, u.dependencia_id, c.nombre
      from alquileres a join lines_items l on (a.id = l.alquiler_id)
      join devoluciones d on (l.id = d.line_item_id)
      join ejemplares e on (l.ejemplar_id = e.id)
      join usuarios u on (a.usuario_id = u.id)
      join dependencias c on (u.dependencia_id = c.id )
      where (a.estatus = 'Alquilado' or a.estatus = 'Alquiler Finalizado') and a.configuracion_id = ?
      group by c.id", params[:id_periodo]])
```

Figura 4.46: Ejemplares entregados a cada escuela

```
#Libros alquilados agrupados por usuarios de las escuelas

    @usuarios_escuela = Ejemplar.find_by_sql(["select count(distinct(u.id)) as cantidad, u.dependencia_id, c.nombre
      from alquileres a join lines_items l on (a.id = l.alquiler_id)
      join devoluciones d on (l.id = d.line_item_id)
      join ejemplares e on (l.ejemplar_id = e.id)
      join usuarios u on (a.usuario_id = u.id)
      join dependencias c on (u.dependencia_id = c.id )
      where (a.estatus = 'Alquilado' or a.estatus = 'Alquiler Finalizado') and a.configuracion_id = ?
      group by c.id", params[:id_periodo]])
```

Figura 4.47: Cantidad de usuarios por escuela

```
@recaudado = Alquiler.find_by_sql(['select sum(e.costo_alquiler) as total
from alquileres a join lines_items l on (a.id = l.alquiler_id)
join ejemplares e on (l.ejemplar_id = e.id)
where (a.estatus = ? or a.estatus = ?)
and a.configuracion_id = ?',"Alquilado", "Alquiler Finalizado", params[:id_periodo]])
```

Figura 4.48: Cantidad de usuarios por escuela

4.1.4. Pruebas

Se muestran las pruebas de esta iteración.

#	DESCRIPCIÓN DEL CASO DE PRUEBAS	RESULTADO ESPERADO	RESULTADO OBTENIDO
1	Ver las gráficas del sistema al terminar el periodo.	Deben mostrar los datos correctos las gráficas al finalizar el periodo cursante.	Se observan las gráficas perfectamente, sin perder datos durante todo el periodo.
2	Ver las gráficas del sistema de periodos anteriores.	Deben mostrar los datos correctos las gráficas de periodos anteriores.	Se pueden seleccionar diferentes periodos y se observan las estadísticas de dicho periodo y el monto recaudado.
3	El contador muestra el monto recaudado al finalizar el periodo.	Mantener el contador del monto recaudado.	Se tiene el contador del monto recaudado durante y al finalizar el periodo.

Cuadro 4.3: Pruebas Iteracion 1

4.2. Iteracion 2: Liberación de libros pre-alquilados y búsqueda de alquileres

4.2.1. Planificación

Se muestran las historias de usuario de esta iteración.

ID	FECHA	DESCRIPCIÓN
5	10-04-2013	Añadir la búsqueda por "código de alquiler" en el área de alquileres del módulo de administración.
6	17-04-2013	Se requiere liberar los libros que fueron pre-alquilados y no terminaron el proceso de alquiler.
7	17-04-2013	Se deben liberar solo los alquileres que tengan mas de tres días en el estado pre-alquilado.

Cuadro 4.4: Historia de Usuario. Iteracion 2.

4.2.2. Diseño

Liberación de alquileres

La solicitud de liberar los alquileres que se encontraban en estado pre-alquilado se resolvió habilitando un enlace, el cual permite que los ejemplares que estén asociados a estos alquileres puedan ser alquilados por otros usuarios. A esta solución se le agrega la restricción que no permite la liberación de alquileres que tengan menos de tres (3) días de creación.

Este procedimiento se trató de implementar en el módulo de administración junto algún otro proceso, para que se ejecutara automáticamente. Sin embargo, no se encontró una ubicación adecuada que permitiera diariamente realizar este procedimiento.

Por esta razón, se estableció un enlace que permite la ejecución del procedimiento de liberación solo cuando el administrador le parezca necesario. Antes de su ejecución existe un formulario de seguridad solo para establecer que el administrador encargado es el único que puede realizar esta operación, esto podemos observarlo en la Figura 4.49.

Figura 4.49: Activación del procedimiento de liberación

4.2.3. Codificación

Busqueda por código de alquiler

Al momento de plantear la solución para agregar una nueva búsqueda de alquileres por medio de su identificador "código de alquiler", se necesitó condicionar el tipo de entrada que obteníamos de la vista; es decir, si la consulta era de tipo numérica y con una cantidad

menor a 5 dígitos la consulta se realizaba por id de alquiler, de lo contrario se mantenía la búsqueda que se realizaba anteriormente. Este sistema de búsqueda en el módulo de administración viene dado por una gema de Ruby, que facilita las consultas en la misma tabla o aquellas que estén relacionadas, permitiendo la búsqueda por caracteres o por números. Sin embargo, realizar la búsqueda por ambos tipos en una misma entrada no se contemplaba.

```
@antes = "#{params[:search]}"
@antes = @antes.sub( "estatus_or_usuario_cedula_or_usuario_nombre_or_usuario_dependencia_nombre_contains", "" )
@antes_lenght = @antes.length
@antes_int = @antes.to_i

if @antes_int != 0 && @antes_lenght <= 4

  sql = "SELECT * FROM alquileres WHERE id = #{@antes_int}"
  @sql_alquileres = Alquiler.find_by_sql(["#{sql}"])
  @alquileres =  Alquiler.paginate_by_sql(sql, :page => params[:page], :per_page => 10)

else

  @search = Alquiler.order('id desc').search(params[:search])
  @alquileres = @search.paginate(:page => params[:page], :per_page => 10)
end
```

Figura 4.50: Búsqueda de alquileres

En la Figura 4.50, podremos observar cómo se realizan las búsquedas por “código de alquiler” y con la gema de Ruby.

Liberación de alquileres

El proceso de liberar alquileres se llevó a cabo por medio de varias consultas SQL que permiten capturar los registros de alquiler que cumplen con la condición de tener más de tres (3) días de creación. Se separa en cuatro pasos, el primer y segundo paso crean las tablas que contienen los alquileres y ejemplares a cambiar, luego dos procedimientos que se encargan

de ejecutar los cambios.

En la Figura 4.51 se muestra la consulta para obtener los alquileres que tengan más de tres (3) días creados y correspondan al último periodo cursado (no hay periodo activo) o el período actual.

```
query1 =  "CREATE TABLE temporal_alquileres AS
          SELECT id FROM `alquileres` WHERE `estatus` LIKE 'prealquilado'
          AND `created_at` < DATE_SUB( NOW( ) , INTERVAL 4  DAY )
          AND `configuracion_id` = #{@periodo_id};"

ActiveRecord::Base.connection.execute(query1);
```

Figura 4.51: Alquileres por liberar

En la Figura 4.52 se muestra la consulta para obtener los ejemplares que estén relacionados con cada uno de los alquileres que se encuentran en la tabla antes creada.

```
query2 =    "CREATE TABLE temporal_ejemplares AS
            SELECT e.id
            FROM lines_items li
            JOIN ejemplares e ON e.id = li.ejemplar_id
            WHERE alquiler_id IN (SELECT id FROM temporal_alquileres);"

ActiveRecord::Base.connection.execute(query2);
```

Figura 4.52: Ejemplares por liberar

En las Figuras 4.53 se cambia los ejemplares a “disponiblez así liberar aquellos que puedan ser alquilados por otros usuarios.

```
query3 =    "UPDATE ejemplares SET estatus_ejemplar = 'Disponible'
            WHERE id IN (SELECT id FROM temporal_ejemplares);"

ActiveRecord::Base.connection.execute(query3);
```

Figura 4.53: Actualización de ejemplres

En la Figura 4.54 y 4.55 se cambia los registros de alquileres y devoluciones al estado "cancelado" para denotar el tipo de procedimiento que se hizo.

```
query4 =  "UPDATE devoluciones SET estatus = 'Cancelado'
          WHERE alquiler_id IN (SELECT id FROM temporal_alquileres);"

ActiveRecord::Base.connection.execute(query4);
```

Figura 4.54: Actualización de registros en devoluciones

```
query5 =  "UPDATE alquileres SET estatus = 'Cancelado'
           WHERE id IN (SELECT id FROM temporal_alquileres);"

ActiveRecord::Base.connection.execute(query5);
```

Figura 4.55: Actualización de alquileres

4.2.4. Pruebas

Se muestran las pruebas de esta iteración.

#	DESCRIPCIÓN DEL CASO DE PRUEBAS	RESULTADO ESPERADO	RESULTADO OBTENIDO
4	Se hace una búsqueda por "código de alquiler" usando un número de 4 dígitos.	Se obtiene solo un resultado de existir el código de alquiler introducido.	Solo se obtiene un resultado que coincide con el número del alquiler.
5	Se hace una búsqueda por nombre, cédula, estatus o escuela.	Se mantiene la funcionalidad del sistema antes del cambio.	El resultado obtenido es el esperado, se verifica probando con los diferentes campos y la consulta arroja la información correcta.
6	Se realizan alquileres y a continuación se busca liberararlos con la nueva funcionalidad.	Los alquileres deberían mantener su estado en "pre-alquilados".	Al seleccionar "liberar libros pre-alquilados", no se modificaron los alquileres que se estaban monitoreando los cuales estaban en el rango de menos de 3 días de creados.
7	Se realizan alquileres y luego de 4 días se liberan.	Los alquileres deben quedar en estado cancelado, y sus ejemplares disponibles.	El sistema realiza los cambios de registros en las tablas ejemplares y alquileres como se esperaba.

Cuadro 4.5: Pruebas Iteracion 2.

4.3. Iteracion 3: Reportes y cambios varios.

4.3.1. Planificación

Se muestran las historias de usuario de esta iteración.

ID	FECHA	DESCRIPCIÓN
8	06-05-2013	Reportes en la sección listado de deudores, el formato debe incluir cédula, apellido, nombre, código de alquiler y período lectivo.
9	06-05-2013	Crear un nuevo reporte como ajustes personalizados.
10	06-05-2013	En la sección libros, cuando se va a agregar un nuevo libro el sistema no lo permite.
11	06-05-2013	En el módulo de reportes en la sección libros disponibles, cambiar orden del formato, primero autor y luego el título. Organizar cotas en orden consecutivo y creciente.
12	06-05-2013	Buscar por autor en ejemplar, agregar estatus “desincorporado” y probar la eliminación de ejemplares.

Cuadro 4.6: Historia de Usuario. Iteracion 3.

4.3.2. Diseño

Al trabajar con la solución de la falla que ocurre en el área de libros del módulo de administración (Cuadro 4.6: ID 10), se encontró que la razón por la cual no se podían almacenar libros en ciertas ocasiones es debido a que el ISBN se había establecido como un

campo obligatorio y no debería serlo, ya que hay libros que no poseen ISBN. Por lo tanto, se resolvió haciendo que este campo no sea obligatorio.

Al trabajar en el área de ejemplares para agregar el campo "autor" (Cuadro 4.6: ID 11) en las búsquedas debíamos extender solamente el uso de las búsquedas complejas que tiene instalado el sistema, en el cual al agregar el campo a buscar en su sintaxis, se logra obtener los resultados deseados al buscar por autor.

La sintaxis que usa esta gema para realizar las búsquedas depende del atributo "name" del campo input de la vista que tiene la estructura siguiente:

name="search[libro_titulo_or_libro_cota_or_libro_autor_or_estatus_ejemplar_contains]"

Figura 4.56: Reportes personalizados

Al trabajar en la nueva tarea que se encarga de generar reportes con datos ajustables (Cuadro 4.6: ID 9) se diseñó una interfaz donde se muestran todas las opciones que tienen

los usuarios para cambiar y obtener los resultados al ejecutar su consulta. Inclusive, si solo quieren visualizar el reporte puede descargarlo en PDF. Para tener una idea más clara de cómo se ve el sistema, observen la Figura 4.56.

La importancia de estos reportes radica en la posibilidad de observar la información más concisa por escuela con datos de libros alquilados y total de ejemplares.

4.3.3. Codificación

Los cambios en los reportes de lista de deudores y libros disponibles (Cuadro 4.6: ID 8) se resolvieron haciendo cambios en las consultas SQL. En la consulta de la Figura 4.57 podemos observar que se está obteniendo el periodo y a su vez el ID del alquiler, para luego transformarlo en XML y generar el reporte con JasperReport, donde la salida es un archivo PDF.

```
@coleccion = Usuario.find_by_sql(
                ["SELECT distinct (u.cedula ), u.nombre,
                        u.dependencia_id, a.id AS alquiler,
                        CONCAT(c.periodo,'-',c.ano ,' ( ', c.tipo_periodo,' )') as periodo,
                FROM usuarios u join alquileres a on (u.id = a.usuario_id)
                JOIN devoluciones d  on (a.id = d.alquiler_id)
                JOIN configuraciones c on (c.id = a.configuracion_id  )
                WHERE d.estatus = 'Sin devolver'
                AND c.id = ? order by u.dependencia_id, u.cedula asc ",params[:id_periodo]]


send_doc( @coleccion.to_xml (:include => { :dependencia => {} }  ),
         '/usuarios/usuario/dependencia',
         'rptDeudores.jasper',
         "Listado de Deudores",
          params[:output_type]
        )
```

Figura 4.57: Consulta SQL lista de deudores

Al realizar los cambios de libros disponibles (Cuadro 4.6: ID 8) se modificó el archivo "rptListadoLibrosDisponibles.jasper" con la ayuda de Ireport que permite diseñar reportes.

La consulta SQL que contiene toda la información de los libros disponibles se observa en la Figura 4.58.

```
def listado_libros_disponibles

        params[:output_type] = "pdf"
        @coleccion = Libro.find_by_sql(['
                SELECT count(e.id) as numero, l.id, l.cota,
                    d.nombre, l.titulo, l.autor, l.edicion,
                    l.ano, e.costo_alquiler
                FROM libros l join dependencias d on (l.dependencia_id = d.id )
                JOIN ejemplares e on (l.id= e.libro_id)
                GROUP BY l.id
                ORDER BY d.nombre, l.cota ']
        )

        send_doc( @coleccion.to_xml, '/libros/libro',
                 'rptListadoLibrosDisponibles.jasper',
                 "Libros Disponibles", params[:output_type]
        )

end
```

Figura 4.58: Consulta SQL de libros disponibles

La tarea de mayor costo en tiempo y esfuerzo de esta iteración fue generar un reporte personalizado (Cuadro 4.6: ID 9), ya que se requería conocer el uso de Ireport y JasperReport, uno para diseñar el reporte y el otro para generar el archivo de salida tipo PDF.

Entre los reportes que posee el sistema algunos funcionan realizando cálculos en código Java dentro del mismo para mostrarlos en el archivo PDF de salida. Los reportes que se realizan de esta manera, consumen mucho tiempo para generarse debido a estos cálculos.

Por esta razón hemos decidido trabajar con consultas SQL anidadas y complejas, pero se obtuvieron los datos necesarios para mostrarse en el reporte, sin necesidad de ningún cálculo.

Para conocer cómo se realiza este proceso, hay que observar la Figura 4.59 y Figura 4.60, que es la muestra de la consulta en los dos casos: cuando se recibe por parámetro la escuela (Figura 4.59) y cuando no (Figura 4.60).

```
if params[:escuela] == nil || params[:escuela] == ""
    @coleccion = Libro.find_by_sql(
        ["SELECT full.*, count(full.libro_id) as total,
                CONCAT(c.periodo,'-',c.ano ,' ( ', c.tipo_periodo,' )') as periodo
          FROM (SELECT e.libro_id, li.autor,li.dependencia_id as escuela_id,
                CONCAT(li.titulo, ' ( Edicion: ', li.edicion, ' )' ) AS titulo,
                        count(e.libro_id) AS alquilados, d.nombre as escuela,
                        a.configuracion_id
                FROM `alquileres` a
                JOIN lines_items l ON l.alquiler_id = a.id
                JOIN ejemplares e ON l.ejemplar_id = e.id
                JOIN libros li ON e.libro_id = li.id
                JOIN dependencias d ON li.dependencia_id = d.id
                WHERE (a.`estatus` LIKE 'alquilado'
                       OR a.`estatus` LIKE 'alquiler finalizado' ) AND a.`configuracion_id` = ?
                group by e.libro_id) full, ejemplares e, configuraciones c

           WHERE full.libro_id = e.libro_id AND full.configuracion_id = c.id
           GROUP BY full.libro_id
           ORDER BY full.#{params[:campo]} #{params[:orden]}
          ",params[:id_periodo]]
      )


    send_doc( @coleccion.to_xml (:include => { :dependencia => {} }  ), '/libros/libro',
              'rptReportcustom.jasper', "Reportes de alquileres", params[:output_type])
else
```

Figura 4.59: Consulta SQL sin filtro

Hay que destacar que para realizar los reportes personalizados y mostrarlos por pantalla se realiza la misma consulta SQL con las mismas condiciones, con la diferencia de que no se

envían los datos para ser transformados en reportes sino que se envían a la vista.

```
else
    @coleccion = Libro.find_by_sql(
       ["SELECT full.*, count(full.libro_id) as total,
            CONCAT(c.periodo,'-',c.ano ,' ( ', c.tipo_periodo,' )') as periodo
         FROM (SELECT e.libro_id, li.autor, li.dependencia_id as escuela_id,
                CONCAT(li.titulo, ' ( Edicion: ', li.edicion, ' )' ) AS titulo,
                count(e.libro_id) AS alquilados, d.nombre as escuela,
                a.configuracion_id
                FROM `alquileres` a
                JOIN lines_items l ON l.alquiler_id = a.id
                JOIN ejemplares e ON l.ejemplar_id = e.id
                JOIN libros li ON e.libro_id = li.id
                JOIN dependencias d ON li.dependencia_id = d.id
                WHERE (a.`estatus` LIKE 'alquilado'
                OR a.`estatus` LIKE 'alquiler finalizado' ) AND a.`configuracion_id` = ?
                group by e.libro_id) full, ejemplares e, configuraciones c
         WHERE full.libro_id = e.libro_id AND full.escuela_id = #{params[:escuela]}
               AND full.configuracion_id = c.id
         GROUP BY full.libro_id
         ORDER BY full.#{params[:campo]} #{params[:orden]}
          ",params[:id_periodo]]
          )


    send_doc( @coleccion.to_xml (:include => { :dependencia => {} }  ), '/libros/libro',
              'rptReportcustom.jasper', "Reportes de alquileres", params[:output_type])
end
```

Figura 4.60: Consulta SQL filtrado por escuela

4.3.4. Pruebas

Se muestran las pruebas de esta iteración.

#	DESCRIPCIÓN DEL CASO DE PRUEBAS	RESULTADO ESPERADO	RESULTADO OBTENIDO
8	Se trata de agregar un libro sin ISBN.	El sistema debe permitir agregarlo.	Efectivamente, el libro fue agregado con exito.

9	Se trata de agregar un libro con ISBN.	El sistema debe permitir agregarlo.	Efectivamente, el libro fue agregado con exito.
10	Se realizan búsquedas por autor en el listado de ejemplares.	Se deben mostrar todos los ejemplares que pertenecen a dicho autor .	El resultado obtenido es el esperado.
11	Se realizan búsquedas por cota, título y estatus en el listado de ejemplares, para verificar que la busqueda basica continua funcional.	Se deben mostrar todos los ejemplares que tienen la cota, el titulo del libro buscado y en su defecto los ejemplares que coincidan con el estatus tipeado.	El resultado obtenido es el esperado.

Cuadro 4.7: Pruebas. Iteracion 3.

Las pruebas de reportes son solo a nivel visual, exceptuando el "Reporte Personalizado"que se toman ciertos libros y se hacen los calculos para corroborar que los datos son correctos.

4.4. Iteracion 4: Historial y cambio de ejemplares.

4.4.1. Planificacion

Se muestran las historias de usuario de esta iteración.

ID	FECHA	DESCRIPCIÓN
13	17-04-2013	Obtener un listado de usuarios que alquilaron un libro.
14	17-04-2013	Cambiar ejemplares antes de alquilar (mientras está en estado pre-alquilado)
15	17-04-2013	Agregar un botón "lo quería" cuando un libro se agota y desarrollar esta operación.

Cuadro 4.8: Historia de Usuario. Iteracion 4.

4.4.2. Diseño:

Cuando un libro se agota, en el catalogo de libros del modulo de estudiantes, se muestra un nuevo botón llamado "lo quería" (Cuadro 4.8: ID 15), el cual ayuda a contabilizar la cantidad de veces que fue requerido este libro después de agotarse, podemos verlo en la Figura 4.61. Ademas, se habilitó un enlace que al seleccionarlo muestre los usuarios que han alquilado dicho ejemplar.

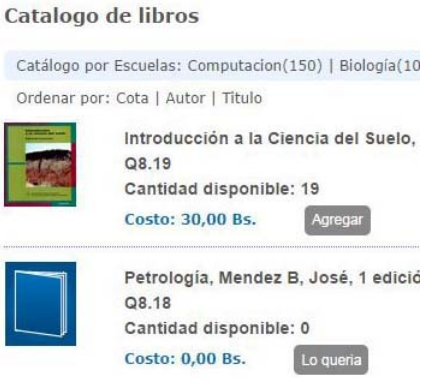


Figura 4.61: Ejemplar agotado con botón "lo quería"

En el área de libros del módulo de administración se muestra la totalidad de la demanda de los libros como un campo, en la Figura 4.62 se puede ver los libros ordenados por cantidad "demanda" y el campo "solicitudes" indica el número total.

Figura 4.62: Listado de libros ordenado por demandado

El cambio de ejemplares (Cuadro 4.8: ID 14)es la opción habilitada en el área de alquileres del módulo de administración, la cual aparece si el alquiler a editar se encuentra en estatus "pre-alquilado".

El usuario podrá cambiar un ejemplar por otro, según lo indique la lista de opciones que se despliega luego que se marque el botón "cambiar", como se observa en la Figura 4.63. Ademas, se agregó un botón para eliminar ejemplares que le permite al administrador remover ejemplares de un alquiler.

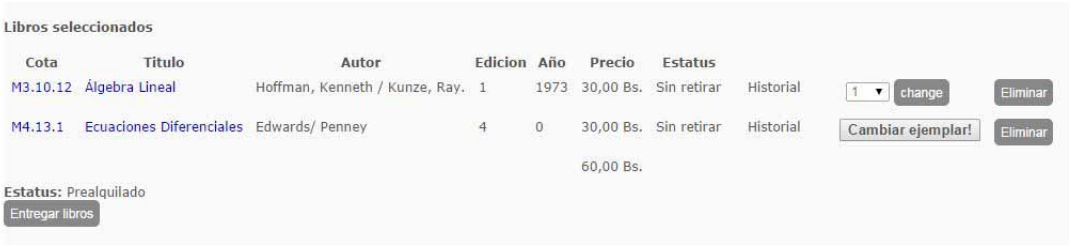


Figura 4.63: Cambio de Ejemplares

En este listado de ejemplares por cambiar, pueden encontrarse ejemplares con estatus disponibles o que pertenecen a otro alquiler con estado “pre-alquilado”, ya que en ambos casos los libros se encuentran todavía en estantería.

Cuando se decide el número del ejemplar por el cual se realizara el cambio, se procede a presionar el botón “cambiar” y este ejemplar se asigna al alquiler que se estaba editando.

4.4.3. Codificación

Para obtener un listado de usuarios que alquilaron un ejemplar (Cuadro 4.8: ID 13) se realiza por medio de una consulta SQL como se muestra la Figura 4.64, donde se usa la tabla que relaciona un alquiler y un ejemplar llamada “line_items”, para obtener los usuarios y al unir la tabla usuarios se obtuvo la información de cada uno.

```
def history_user_ejemplar

  @preview_url = params[:preview_url]

  params[:ejemplar_id]

  sql = "SELECT u.*
         FROM `alquileres` a
         JOIN `lines_items` li on a.id = li.alquiler_id
         JOIN `usuarios` u on a.usuario_id = u.id
         WHERE li.ejemplar_id = #{params[:ejemplar_id]} ORDER BY alquiler_id DESC
       "

  @page = params[:page]
  @usuarios =  Usuario.paginate_by_sql(sql, :page => @page, :per_page => 10)

end
```

Figura 4.64: Consulta de historial de usuarios que alquilaron un ejemplar

Cada vez que un usuario selecciona el botón “lo quería” (Cuadro 4.8: ID 15)se dispara un evento en Jquery el cual se encarga de ejecutar la petición de sumar una unidad mas al campo demanda que se almacena en la tabla de libros. El evento Jquery es el que observamos en la Figura 4.65, el cual ejecuta un procedimiento llamado “increment_likes”.

```
$("html").delegate(".loqueria","click",function(e) {
    var libro_id = $(this).attr("libro_id");
    e.preventDefault();
    $.post("/ppal_estudiante/increment_likes", {'id':libro_id},function(data){

        $(".loqueria").hide();
        return false;
    });

});
```

Figura 4.65: Evento Jquery para función “lo quería”

El método que incrementa en una unidad el campo demanda de la tabla libro, como se muestra en la Figura 4.66.

```
def increment_likes
    @libro = Libro.find(params[:id])

    @libro.demanda = @libro.demanda + 1
    if @libro.save
      @value = 1
    else
      @value = 0
    end
    render :nothing => true
end
```

Figura 4.66: Agregar +1 al campo demanda

Cuando se busca procesar el cambio de un ejemplar (Cuadro 4.8: ID 14), se recibe el parámetro de la vista que indica el número de ejemplar seleccionado para cambiar y dependiendo de la condición de éste se realizará uno u otro proceso. Si el estatus del ejemplar era "disponible", se procede a cambiar nuestro ejemplar a "disponible" y el seleccionado pasa a "alquilado". Si es el otro caso, el cambio se realiza a nivel de alquileres, nuestro alquiler lo relacionamos con el ejemplar seleccionado y el alquiler que se relacionaba con este, pasa a nuestro ejemplar. Hay que recordar que se habla de "nuestro alquiler" porque este procedimiento se está realizando en la edición de un alquiler en el módulo de administración.

```
# Inicia cambio de ejemplar viejo por nuevo
  @ejemplar_current = Ejemplar.find(params[:id_ejemplar_current])
  @ejemplar_new = Ejemplar.find(params[:select_ejemplar])

  @temp_ejem_curr = @ejemplar_current.dup
  @temp_ejem_new = @ejemplar_new.dup
  @ejemplar_current.estatus_ejemplar = @ejemplar_new.estatus_ejemplar
  @ejemplar_new.estatus_ejemplar = @temp_ejem_curr.estatus_ejemplar

  @ejemplar_new.save
  @ejemplar_current.save

  guardar_log(session[:usuario_id], self.class.name,__method__.to_s,
              @temp_ejem_curr,nil )
  guardar_log(session[:usuario_id], self.class.name,__method__.to_s,
              @temp_ejem_new,nil )
# Fin cambio de ejemplar viejo por nuevo

redirect_to "/alquileres/#{params[:id_alquiler]}/edit?locale=es"
```

Figura 4.67: Cambio de estatus de ejemplares

En la Figura 4.67 y 4.68, mostramos la consulta que acabamos de explicar. En la cual "line_item" es la tabla relación existente entre un alquiler y un ejemplar.

```
@lin_items_current = LineItem.find(params[:id_line_current])
@lin_items_n = LineItem.where("ejemplar_id = :ejemplar_id",
                {:ejemplar_id => params[:select_ejemplar]})

# Inicia cambio de line_item viejo por nuevo
  if(@lin_items_n.count != 0)
      @lin_items_new = LineItem.find(@lin_items_n[0].id)
      @lin_items_new.ejemplar_id = @lin_items_current.ejemplar_id

      @temp_new = @lin_items_new.dup
      @lin_items_new.save
      guardar_log(session[:usuario_id], self.class.name,__method__.to_s,
                  @temp_new,nil )
  end

  @lin_items_current.ejemplar_id = params[:select_ejemplar]

  @temp_curr = @lin_items_current.dup
  @lin_items_current.save

  guardar_log(session[:usuario_id], self.class.name,__method__.to_s,
              @temp_curr,nil )

  # Fin cambio de line_item viejo por nuevo
```

Figura 4.68: Cambio de registros relacionados

Si existe un caso donde el administrador toma la decisión de eliminar un ejemplar, el sistema comienza cambiando el ejemplar a un estatus "disponible", permitiendo que pueda ser adquirido por otro usuario para ser alquilado, se modifican los registros de las tablas devolución, alquileres y lineitems como se muestra en la Figura 4.69, para completar el proceso.

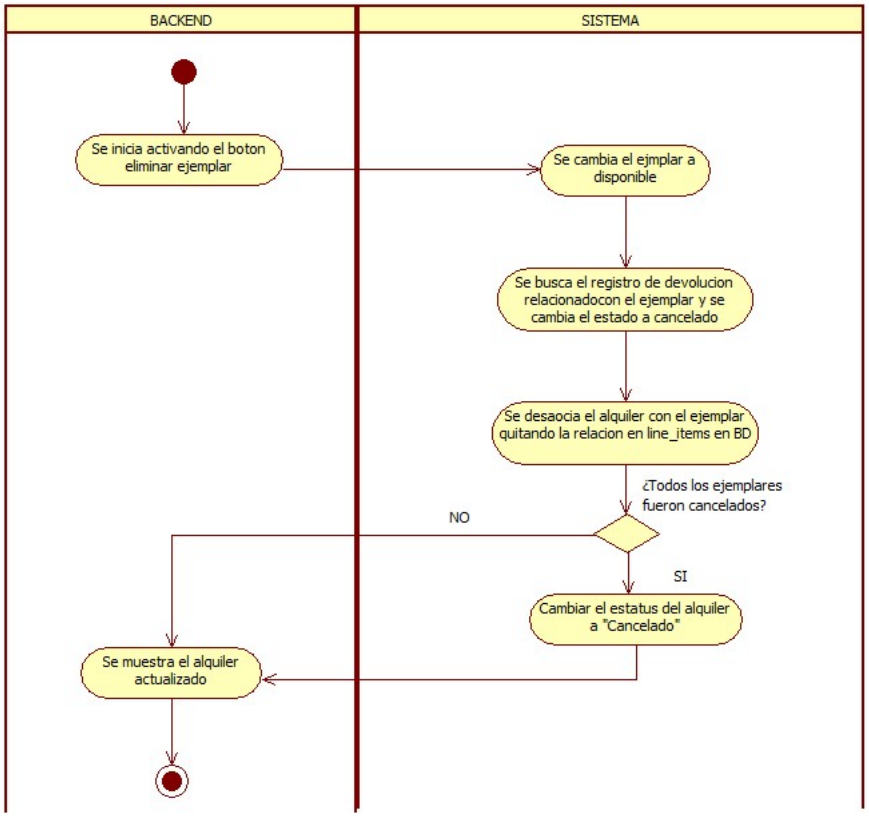


Figura 4.69: Eliminar Ejemplares en Alquileres

4.4.4. Pruebas

Se muestran las pruebas de esta iteración.

#	DESCRIPCIÓN DEL CASO DE PRUEBAS	RESULTADO ESPERADO	RESULTADO OBTENIDO
12	Listar un historial de alquiler de un libro.	Los ejemplares mostraran los usuarios que han alquilado dicho ejemplar.	Se muestran todos los usuarios que han alquilado dicho libro.
13	Cambiar un ejemplar mientras esta pre-alquilado. Seleccionar otro ejemplar pre-alquilado.	Al seleccionar otro ejemplar pre-alquilado se intercambian solo los ejemplares.	Se obtuvo el resultado esperado, ya que se hizo seguimiento a los registros en la tabla "line_item" de la BD.
14	Cambiar un ejemplar mientras esta pre-alquilado. Seleccionar un ejemplar disponible.	Al seleccionar cambio de ejemplar se habilita el listado de otros ejemplares del mismo libro que estén pre-alquilados o disponibles (en ambos casos están en estantería). Al usar un ejemplar disponible	Se cambió por un ejemplar disponible. El viejo ejemplar cambia su estado de pre-alquilado a disponible.
15	Mostrar el botón "lo quería"	Debe aparecer el botón "lo quería" y al presionarlo incrementar el campo demanda .	Apareció el botón y al presionarlo en base de datos se sumo +1 al campo demanda.

Cuadro 4.9: Pruebas. Iteracion 4.

4.5. Iteración 5: Roles y Solvencias.

4.5.1. Planificación

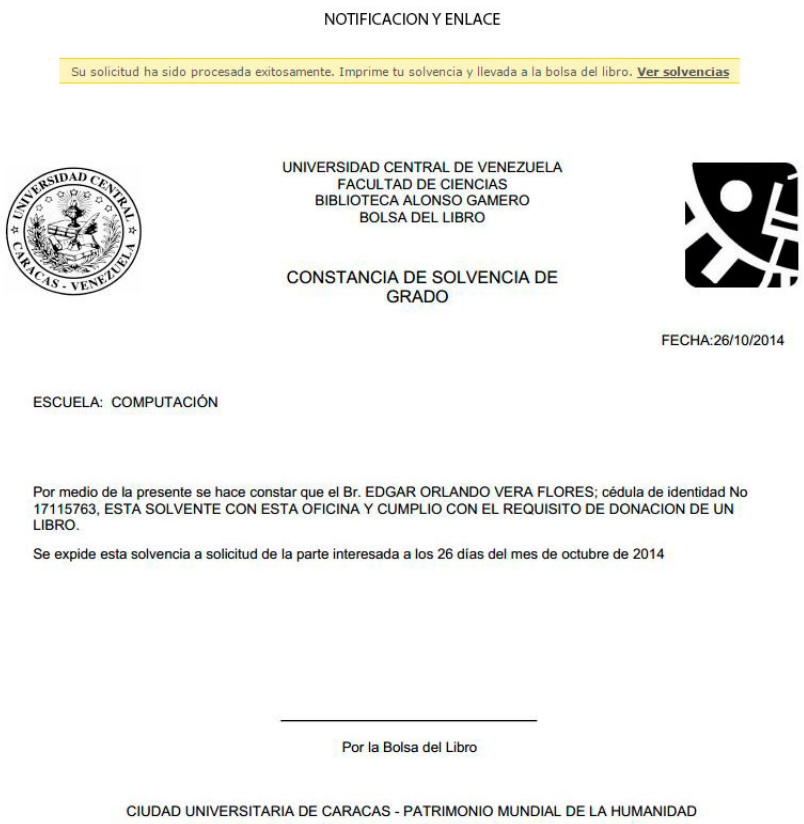
Se muestran las historias de usuario de esta iteración.

ID	FECHA	DESCRIPCIÓN
16	23-05-2013	Establecer prioridades o roles en el módulo de administración para poder visualizar diferentes funcionalidades según el tipo de rol.
17	23-05-2013	En el comprobante de alquiler que genera el sistema cuando el usuario realiza un alquiler debería contener la escuela a la que pertenece el usuario.
18	04-06-2013	Hay un error y se permite solicitar solvencias a los usuarios teniendo libros alquilados.
19	04-06-2013	Cuando el estudiante emita la solvencia indicarle que se realizó la solicitud y darle la opción a ver el PDF que llevará a la bolsa del libro.

Cuadro 4.10: Historia de Usuario. Iteracion 5.

4.5.2. Diseño

Al realizar cambios en las solvencias para no permitir solicitarlas si se deben libros (Cuadro 4.10: ID 18), también se añadió la “escuela” en la constancia y al final una ayuda visual, donde se muestra un enlace donde tendrá acceso a la constancia de solvencia en PDF (Cuadro 4.10: ID 19), como lo muestra la Figura 4.70.


NOTIFICACION Y ENLACE

Su solicitud ha sido procesada exitosamente. Imprime tu solvencia y llevada a la bolsa del libro. **Ver solvencias**

UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
BIBLIOTECA ALONSO GAMERO
BOLSA DEL LIBRO

CONSTANCIA DE SOLVENCIA DE GRADO

FECHA:26/10/2014

ESCUELA: COMPUTACIÓN

Por medio de la presente se hace constar que el Br. EDGAR ORLANDO VERA FLORES; cédula de identidad No 17115763, ESTA SOLVENTE CON ESTA OFICINA Y CUMPLIO CON EL REQUISITO DE DONACION DE UN LIBRO.

Se expide esta solvencia a solicitud de la parte interesada a los 26 días del mes de octubre de 2014

Por la Bolsa del Libro

CIUDAD UNIVERSITARIA DE CARACAS - PATRIMONIO MUNDIAL DE LA HUMANIDAD

Figura 4.70: Notificación, enlace y constancia solicitada.

Para establecer prioridades de roles y privilegios (Cuadro 4.10: ID 16), se agregó un nuevo campo en el área de usuarios del módulo de administración. Este nuevo campo se habilita cuando se selecciona en rol la opción "administrador" y despliega 2 opciones a elegir: observador o privilegiado, como se muestra en la Figura 4.71.

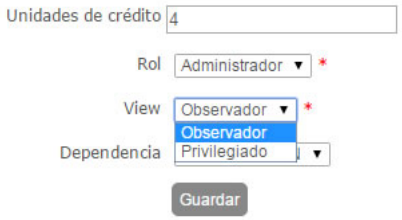


Figura 4.71: Extensión de roles administración.

Por los momentos, los administradores de tipo "observador" no tienen opción de editar ni eliminar en ningún área del módulo de administración, excepto en el área de solvencias donde tiene la posibilidad de marcar como entregado las solvencias, como se observa en la Figura 4.72.

Figura 4.72: Vista administrador "Observador"

4.5.3. Codificación

Primero analizaremos el cambio en el comprobante de alquiler (Cuadro 4.10: ID 17), donde se agrega la escuela en la consulta que genera el reporte y en el diseño del mismo se agrega el lugar donde aparecerá al crearse el PDF.

Luego se establecieron las restricciones al momento de solicitar solvencias en el módulo de estudiantes (Cuadro 4.10: ID 18), donde se consulta si existe algún libro sin devolver como se observa en la Figura 4.73.

```
def es_solvente(usuario_id)
            @configuracion = current_config

  @last_periodos = Configuracion.where("id >= :id", {:id => 1}).order('id DESC').take(1)

  if(@configuracion.blank?)
    @configuracion = @last_periodos
  end

            @coleccion = Usuario.find_by_sql(['select distinct (u.cedula ),
                                              u.nombre, u.dependencia_id
            from usuarios u join alquileres a on (u.id = a.usuario_id)
            join devoluciones d  on (a.id = d.alquiler_id)
            where d.estatus = "Sin devolver"
                and u.id = ? order by u.dependencia_id, u.cedula asc ',usuario_id])
      if @coleccion.count == 0
            return true
      else
            return false
      end

end
```

Figura 4.73: Función de solvencia

Se hace una llamada a esta función que retorna “verdad” o “falso” como condición para de guardar o no la solicitud de solvencia, como se observa en la Figura 4.74.

```
def guardar_solicitud_solvencia
      @temp = Solvencia.where(:usuario_id => session[:usuario_id], :estatus => "Solicitud Enviada" )

      if es_solvente(session[:usuario_id])== true and @temp.count == 0
              @solvencia = Solvencia.new
              @solvencia.estatus = "Solicitud Enviada"
              @solvencia.tipo_solvencia = params[:tipo_solvencia]
              @solvencia.usuario_id = session[:usuario_id]
              if @solvencia.save
              guardar_log(session[:usuario_id], self.class.name,__method__.to_s, @solvencia,nil )
              session[:solvencia_id] = @solvencia.id
                      redirect_to ppal_estudiante_index_path,
                          :notice => "Su solicitud ha sido procesada exitosamente.
                           Imprime tu solvencia y llevada a la bolsa del libro."
              else
                      redirect_to ppal_estudiante_index_path,
                          :notice => "La solicitud no pudo ser procesada, intente mÃ¡s tarde"
              end

      else
              redirect_to ppal_estudiante_index_path,
                          :notice => "Usted ya posee una solvencia en proceso"
      end

end
```

Figura 4.74: Método para guardar solicitud

Al final de esta iteración se analiza el proceso de creación de roles (Cuadro 4.10: ID 16), donde podemos verlo como una variación del modelo de usuarios permitiendo recibir una variable mas que es el campo “vista”, la cual servirá para definir el tipo de privilegio que tienen los administradores.

El proceso para implementar los privilegios de usuario en el módulo de administración, inicia cundo el usuario se autentica, se guarda la variable que indica sus privilegios de acceso y por medio de Jquery se deshabilitan las opciones de edición y eliminación, con el evento que se observa en la Figura 4.75.

```
var session_rol = $('.hidden_panel #user_rol').val();
if(session_rol == 'Administrador'){
    var session_view = $('.hidden_panel #user_view').val();
    if(session_view != 'Privilegiado'){

        $('.view_edit').remove()
    }
}
```

Figura 4.75: Evento eliminar opción de editar o eliminar

Además, para restringir a nivel de servidor, se condiciona las acciones de "edit" y "delete" para ser usadas solo por administradores de tipo "Privilegiado" con la sentencia que se muestra en la Figura 4.76.

```
def edit
  if session[:view] != 'Privilegiado'
      redirect_to areas_conocimientos_url, :notice => 'No tienes permisos para editar'
  end

  add_breadcrumb "Editar alquiler", :edit_alquiler_path
  @alquiler = Alquiler.find(params[:id])
    @aux = Array.new(@alquiler.line_item.count, Hash.new)
    @alquiler.line_item.each do |items|

      @all_ejemplares = Ejemplar.where("libro_id = :libro_id
          and (estatus_ejemplar = :estatus_ejemplar1 or estatus_ejemplar = :estatus_ejemplar2)",
           {:libro_id => items.ejemplar.libro_id, :estatus_ejemplar1 => "Disponible",
           :estatus_ejemplar2 => "Prealquilado"})
      @aux[items.ejemplar.id] = @all_ejemplares

    end

end
```

Figura 4.76: Condicional para edición

4.5.4. Pruebas

Se muestran las pruebas de esta iteración.

#	DESCRIPCIÓN DEL CASO DE PRUEBAS	RESULTADO ESPERADO	RESULTADO OBTENIDO
16	Probar roles de administración para usuarios de tipo "Observador"	Las funcionalidades de este usuario deberían ser mínimas y solo poder manipular el área de solvencias.	Se logra mantener igualdad de funcionalidades en el área de solvencias, para las demás funciones en el módulo de administración solo tiene permiso de observar, no puede modificar ni eliminar.
17	Probar roles de administración con todos los permisos.	Todas las funcionalidades de los administradores con todos los derechos deben estar permitidas y activas.	Una vez establecido el sistema de roles se probó las funcionalidades de los administradores que tienen todos permisos "privilegiado" y funcionan perfectamente.
18	Realizar una solvencia como estudiante con "libros alquilados"	Al intentar realizar la solicitud de solvencia, no se permite.	Se hizo la prueba de solicitar una solvencia teniendo libros sin entregar, se notifica que no se puede realizar la solicitud por no estar solvente.
19	Realizar una solvencia como estudiante sin "libros alquilados"	Al intentar realizar la solicitud de solvencia, se permite realizarla.	Se permitió realizar la solvencia, se mostró la notificación de aprobación y el enlace para visualizar la constancia en PDF.

Cuadro 4.11: Pruebas. Iteracion 5.

4.6. Iteracion 6: Creación del módulo de profesores.

4.6.1. Planificación

Se muestran las historias de usuario de esta iteración.

ID	FECHA	DESCRIPCIÓN
20	21-10-2013	Crear módulo de profesores donde se establezcan diferentes roles.
21	21-10-2013	Crear y administrar las sugerencias de libros de profesores.

Cuadro 4.12: Historia de Usuario. Iteracion 6.

4.6.2. Diseño

El proceso de creación de roles (Cuadro 4.12: ID 20), es una variación del modelo de usuarios donde el campo "vista" permitirá identificar el tipo de profesor: normal o coordinador. Véase Figura 4.77.

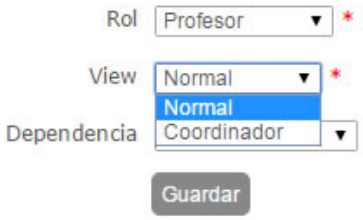


Figura 4.77: Extensión de roles profesores.

Cuando un usuario logra autenticarse como profesor se le despliega una vista y un

menú donde podrá revisar sugerencias y/o visualizar la lista de libros que se encuentran en la Bolsa del Libro, como se observa en la Figura 4.78.

Figura 4.78: Inicio del Módulo de Profesores

Para crear las sugerencias (Cuadro 4.12: ID 21)se diseñó la tabla "sugerencias.[en] la BD con la estructura que se muestra en la Figura 4.79.

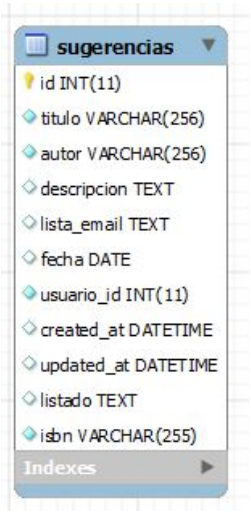


Figura 4.79: Estructura de BD: Tabla sugerencias

Para listar, crear, editar o eliminar sugerencias se diseñaron las vistas que dan acceso a cada una de estas operaciones. El listado de sugerencias se puede ver en la Figura 4.80, mientras que el formulario de creación o edición se observa en la Figura 4.81, que permite guardar el ISBN, titulo, autor y una breve descripción del libro a sugerir.

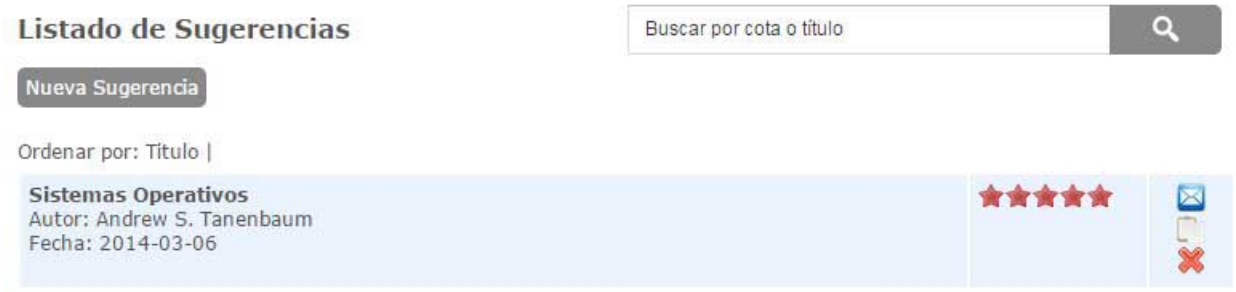


Figura 4.80: Listado de sugerencias

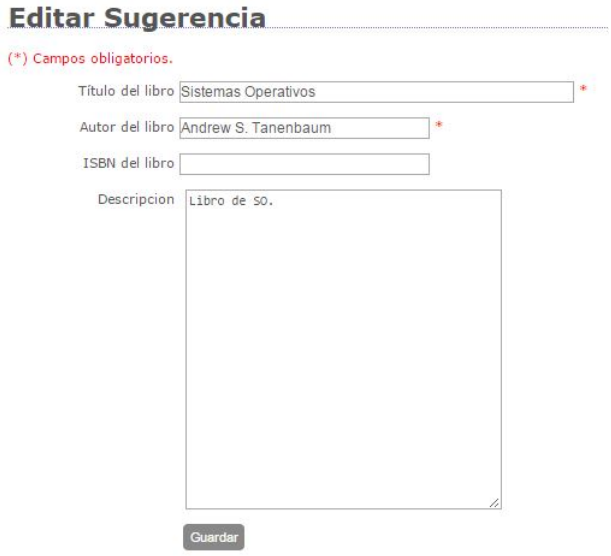


Figura 4.81: Formulario de sugerencia

4.6.3. Codificación

Para establecer el módulo de administración (Cuadro 4.12: ID 20) se creó un nuevo controlador "ppal_profesores_admin_controller" que se encarga de gestionar el flujo de información que pasa a las vistas y los procesos que se pueden ejecutar en este módulo.

Para acceder al módulo de profesores se modificó el proceso que se encuentra en el controlador de "sesión" el cual está encargado de redirigir a los estudiantes, administradores y ahora a los profesores a sus módulos únicamente. El cambio en este proceso se puede observar en la Figura 4.82, que refleja las condiciones que existen para manejar los roles y privilegios de acceso a los módulos.

```
if usuario and usuario.authenticate(params[:password]) and usuario.rol != 'Inactivo'

        session[:usuario_id] = usuario.id
        session[:usuario_nombre] = usuario.nombre
        session[:rol] = usuario.rol
        session[:view] = usuario.view
        session[:expires_at] = 1.minutes.from_now

        if usuario.rol == 'Administrador'
            redirect_to ppal_admin_index_url
        elsif usuario.rol == 'Operador'
            redirect_to ppal_admin_url
        elsif usuario.rol == 'Profesor'
            redirect_to ppal_profesores_admin_index_url
        else
            redirect_to ppal_estudiante_index_url
        end
end
```

Figura 4.82: Codigo de redirección según el rol

Con el uso de "Scaffolding" de Rails se generó todo el código necesario para listar, mostrar, crear, editar o eliminar las sugerencias (Cuadro 4.12: ID 21). Esta instrucción genera el modelo, el controlador, las vistas, estructura en la BD y toda la configuración necesaria para

tener acceso a las acciones y poder almacenar las sugerencias realizadas por sistema.

En la vista que despliega la lista de sugerencias se restringieron los permisos para editar y eliminar sugerencias, si la misma no te pertenece.

Los métodos que permitirán el manejo de sugerencias se pueden apreciar en las Figuras 4.83, 4.84y 4.85. Los cuales son los encargados de crear, editar o eliminar una sugerencia en la BD.

```
def create
  @sugerencia = Sugerencia.new(params[:sugerencia])
  @sugerencia.usuario_id = session[:usuario_id]
  @sugerencia.fecha = Time.now

  if !params[:select_libro].empty?
    redirect_to new_sugerencia_path, :notice => 'El libro sugerido ya existe' and  return
  end

  if params[:sugerencia]['titulo'].empty?
    redirect_to new_sugerencia_path, :notice => 'El titulo no puede ser vacio' and  return
  end

  if params[:sugerencia]['autor'].empty?
    redirect_to new_sugerencia_path, :notice => 'El autor no puede ser vacio' and  return
  end

  respond_to do |format|
    if @sugerencia.save
              guardar_log(session[:usuario_id], self.class.name,__method__.to_s, @sugerencia,nil )
      format.html { redirect_to sugerencias_url, :notice => 'Sugerencia creada exitosamente.' }
      format.json { render :json => @sugerencia, :status => :created, :location => @sugerencia }
    else
      format.html { render :action => "new" }
      format.json { render :json => @sugerencia.errors, :status => :unprocessable_entity }
    end
  end
```

Figura 4.83: Código de creación de sugerencias

```
def update
  @sugerencia = Sugerencia.find(params[:id])
  @sugerencia.usuario_id = session[:usuario_id]
  @sugerencia.fecha = Time.now
      @temp = @sugerencia.dup
  respond_to do |format|
    if @sugerencia.update_attributes(params[:sugerencia])
              guardar_log(session[:usuario_id], self.class.name,__method__.to_s, @temp,@sugerencia  )

      format.html { redirect_to sugerencias_url, :notice => 'Sugerencia actualizado exitosamente.' }
      format.json { head :ok }
    else
      format.html { render :action => "edit" }
      format.json { render :json => @sugerencia.errors, :status => :unprocessable_entity }
    end
  end
end
```

Figura 4.84: Código de actualización de sugerencias

```
  def enviar_email
      @materias = Materia.find(params[:materia_id])
      @listamaterias = Listamaterias.find(params[:id])

      @array_emails = @listamaterias.listado.to_s.split(",")

      if @array_emails.empty?
          mensaje = "No hay una lista de emails para enviar. Agregue email(s) e intente denuevo."

      else
          @materia_libro = Materia.find_by_sql(['SELECT l.cota, l.titulo
                                                 FROM materia_libro ml
                                                 JOIN libros l ON ml.libro_id = l.id
                                                 WHERE materia_id = ?' , params[:materia_id]])

          @aux = ""
          @array_emails.each do |email|
                @text_email = @aux + ", #{email}";

          end


          ClaveMailer.envio_masivo(@array_emails[0],@array_emails,@materia_libro,@materias.nombre).deliver
```

Figura 4.85: Código de eliminación de sugerencias

4.6.4. Pruebas

Se muestran las pruebas de esta iteración.

#	DESCRIPCIÓN DEL CASO DE PRUEBAS	RESULTADO ESPERADO	RESULTADO OBTENIDO
20	Autenticarse como profesor y mostrar el módulo de profesores.	Apenas ingreses al sistema como profesor debes poder ver el módulo de profesores.	Al ingresar al sistema como profesor se observan funcionalidades de administración como profesor.
21	Se busca la manera de ingresar al módulo de estudiantes o administración estando autenticado como profesor.	No se puede acceder a funcionalidades de módulos que no sea el de profesores.	Se autenticó como profesor, y luego por medio de enlaces, no se logró acceder a funcionalidades que no pertenecen al módulo de profesores.
22	Creación, edición, eliminación de sugerencias	Se debe poder crear una sugerencia y solo puede editar y/o eliminar sugerencias de el mismo.	Se obtienen los resultados esperados, donde no se permiten modificar sugerencias que no sean propias.

Cuadro 4.13: Pruebas. Iteracion 6.

4.7. Iteracion 7: Materias en la Bolsa del Libro.

4.7.1. Planificación

Se muestran las historias de usuario de esta iteración.

ID	FECHA	DESCRIPCIÓN
22	13-11-2013	Crear un listado de materias que se dictan en toda la Facultad.
23	13-11-2013	Agregar a las materias una lista de libros sugeridos en el curso y se encuentren en la Bolsa del Libro.
24	13-11-2013	Crear listas de correos de estudiantes a los cuales se les pueda enviar los libros sugeridos por materias.

Cuadro 4.14: Historia de Usuario. Iteracion 7.

4.7.2. Diseño

Se crea una tabla donde se guardan todas las materias que se dictan en la Facultad de Ciencias, para esto se diseñó la estructura de la BD (Cuadro 4.14: ID 22), como se muestra en la Figura 4.86.

Figura 4.86: Estructura de BD: Tabla materias

Para listar, crear, editar o eliminar materias (Cuadro 4.14: ID 22) se diseñaron las vistas que dan acceso a cada una de estas operaciones.

El listado de materias se puede ver en la Figura 4.87, mientras que el formulario de creación o edición se observa en la Figura 4.88, que permite guardar el código, nombre y escuela. Además, se visualiza un campo llamado "libro" (Cuadro 4.14: ID 23) en el cual se va escribiendo el título, el autor o la cota de un libro y se despliega una lista de los libros que coincidan con lo que se está escribiendo. Este listado tiene como fuente la Bolsa del Libro, eso significa que si no se selecciona un libro del listado que aparece, este no será guardado.

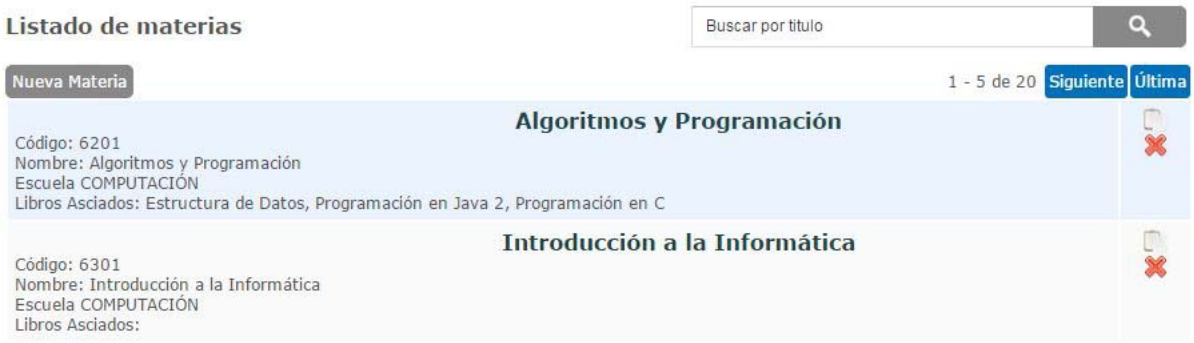


Figura 4.87: Listado de materias

Una vez se tiene seleccionado el libro, se presiona el botón "agregar" y se mostrara el mismo mas abajo (se puede eliminar si ya no se desea agregar este libro). Para finalizar la creación o edición de la materia, se procede a guardar.

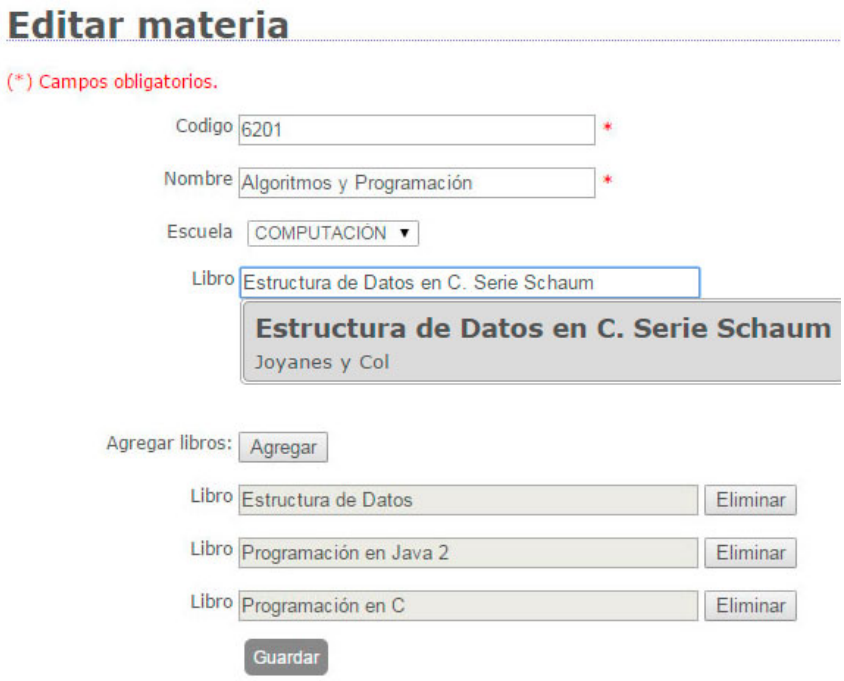


Figura 4.88: Editar Materia

Luego de crear las materias y una lista de libros, se procede a guardar una lista de estudiantes para enviar vía correo electrónico (Cuadro 4.14: ID 24) la información recolectada en las materias, por lo cual se desarrolló la estructura completa (tabla en la BD, modelo, controladores y vistas) que permiten almacenar la lista de correos electrónicos de aquellas personas a las cuales se quiere enviar los enlaces de los libros sugeridos dado una determinada materia.

Se agregó esta nueva opción al menú de profesores y se pueden observar la listas creadas para difundir la información de una materia, como se muestra en la Figura 4.89, que también refleja las operaciones que se pueden realizar con cada lista (crear, editar, eliminar) y aparecerá la opción de “enviar email” cuando la lista tenga por lo menos una dirección de

correo electrónico guardada.

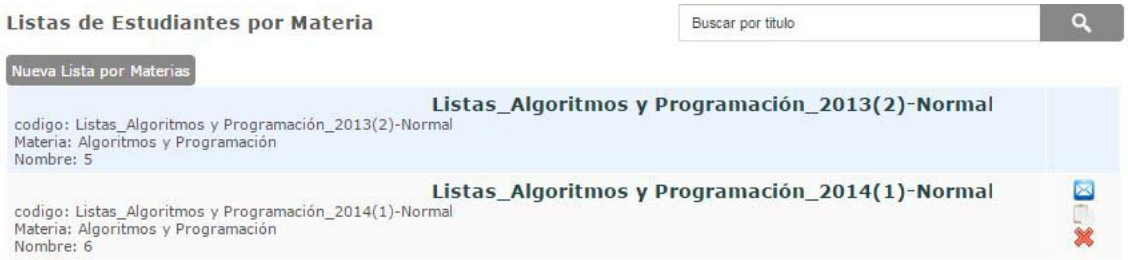


Figura 4.89: Lista de contactos por materias

En la Figura 4.90 podemos observar el formulario que permite asignar la materia y agregar la lista de correos separada por comas (,).

Figura 4.90: Formulario de envío de información de materias

4.7.3. Codificación

La creación de materias y la lista de correos se hicieron mediante el uso de "Scaffolding" de Rails. Con esto tenemos la estructura básica para listar, mostrar, crear, editar y eliminar materias o listas de correos. Sin embargo, para almacenar el listado de libros se realizaron diferentes procesos que permitían listar, agregar o eliminar los libros dentro del formulario de materias.

```
$("html").delegate(".project","keyup",function() {

    var contador = parseInt($('#field_select_libro').attr("contador"));
    var zona = $('#field_select_libro').attr("zona");
    //alert(zona);
    var at1 = '#project-'+contador;
    var at2 = '#project-id-'+contador;
    var at3 = '#project-description-'+contador;
    //alert(at1);


        $.get("/ajax/autocomplete_books", {'id':1},function(databooks) {
            //alert(databooks); return false;
            //console.log(databooks);
            datacomplete(at1, at2, at3, databooks);
        });

});
```

Figura 4.91: Función que lista libros JavaScript

Por medio de Jquery capturamos el evento mientras se va escribiendo en el campo "libro" (Cuadro 4.14: ID 23) del formulario de materias. En la Figura 4.91 se hace una llamada a una función "datacomplete" que se encarga de obtener la lista de libros de la Bolsa del Libro como se muestra en la Figura 4.92 gracias al método "autocomplete_books" que se define en la Figura 4.93.

```
function datacomplete(at1, at2, at3, arreglo){

    $( at1 ).autocomplete({
        minLength: 0,
        max: 10,
        source: arreglo,
        focus: function( event, ui ) {
            $( at1 ).val( ui.item.label );
            return false;
        },
        select: function( event, ui ) {
            $( at1 ).val( ui.item.label );
            $( at2 ).val( ui.item.value );
            $( at3 ).html( ui.item.descrip );

            return false;
        }
    })
    .data( "autocomplete" )._renderItem = function( ul, item ) {
        return $( "<li ></li>" )
        .data( "item.autocomplete", item )
        .append( "<a><div style='font-weight: bold;'>"
                + item.labelhidden +
            "</div><div style='font-size: 13px;'>"
                + item.descrip +
            "</div></a>" )
        .appendTo( ul );
    };
}
```

Figura 4.92: Función que lista libros de BD

```
def listado_libros_disponibles

        params[:output_type] = "pdf"
        @coleccion = Libro.find_by_sql(['
                SELECT count(e.id) as numero, l.id, l.cota,
                    d.nombre, l.titulo, l.autor, l.edicion,
                    l.ano, e.costo_alquiler
                FROM libros l join dependencias d on (l.dependencia_id = d.id )
                JOIN ejemplares e on (l.id= e.libro_id)
                GROUP BY l.id
                ORDER BY d.nombre, l.cota ']
        )

        send_doc( @coleccion.to_xml, '/libros/libro',
                 'rptListadoLibrosDisponibles.jasper',
                 "Libros Disponibles", params[:output_type]
        )

end
```

Figura 4.93: Función que lista libros de BD

Una vez seleccionado el libro se procede a ejecutar el botón "agregar" que por medio de Jquery permite añadir la cantidad de libros que se seleccionen, para ser almacenados en la tabla "materia_libro" que tiene la estructura mostrada en la Figura 4.94.

Figura 4.94: Tabla relación entre libro y materia

En la creación o edición de materias el botón de "agregar" solo añade un campo de libro en el formulario, solo al guardar los cambios de materias se modifican los registros de la tabla materia_libro. La función del botón agregar la podemos ver en la Figura 4.95.

```
$('.add_many_libros').click(function(e) {
    e.preventDefault();
    var contador = parseInt($('#field_select_libro').attr("contador"));
    var suma = contador + 1;

    var full_add = $('#inside_select_libro .project').val();
    if(!(full_add)){
        alert('Debes seleccionar un libro'); return false;
    }

    var $newdiv = $('#inside_select_libro').clone();
    //alert($newdiv);
    $("#project-1").attr("id", "project-"+suma);
    $("#project-id-1").attr("id", "project-id-"+suma);
    $("#project-description-1").attr("id", "project-description-"+suma);
    $("#remove_many_libros-1").attr("id", "remove_many_libros-"+suma);

    $(".input_many_libros").append($newdiv);
    $('#field_select_libro').attr("contador", suma);
    $("#field_select_libro input").val("");

    return false;
});
```

Figura 4.95: Código para agregar libros en materias

En el área de lista de correos por materias del módulo de profesores, se creó la acción "envío de correos" (Cuadro 4.14: ID 24), la cual permite obtener la materia seleccionada y de los libros relacionados crea un enlace a la Bolsa del Libro por cada uno y el contenido del email, se puede ver como la Figura 4.96.

Figura 4.96: Contenido del correo electrónico

4.7.4. Pruebas

Se muestran las pruebas de esta iteración.

#	DESCRIPCIÓN DEL CASO DE PRUEBAS	RESULTADO ESPERADO	RESULTADO OBTENIDO
23	Creación, edición, eliminación de materias que se dictan en la facultad.	Debe poderse crear, editar y eliminar materias.	Se logra crear materias, luego editarlas. Al final se permite eliminarla.

24	Agregar o eliminar libros que se asocian a cada materia.	Se debe poder agregar y eliminar libros que se asociaron a una materia.	En el formulario de creación o edición de una materia se permite agregar una lista de libros y guardarlos. Si nuevamente se editan se logró agregar más libros a la lista e inclusive eliminar libros que ya estaban añadidos.

Cuadro 4.15: Pruebas. Iteracion 7.

4.8. Iteracion 8: Comentarios por sugerencia.

4.8.1. Planificacion

Se muestran las historias de usuario de esta iteración.

ID	FECHA	DESCRIPCIÓN
25	13-11-2013	Se deben poder generar comentarios de las sugerencias hechas por los profesores, solo deben realizarlas otros profesores.

Cuadro 4.16: Historia de Usuario. Iteracion 8.

4.8.2. Diseño

Se crea una tabla donde se guardan todos los comentarios realizados al visualizar el detalle de una sugerencia, para esto se diseñó la estructura de la BD, como se muestra en la Figura 4.97.

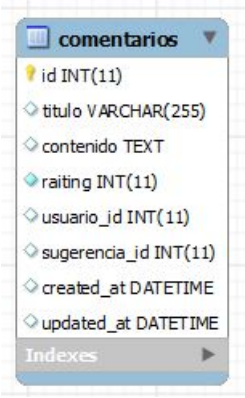


Figura 4.97: Estructura de BD. Tabla comentarios

Tal como se muestra en la Figura 4.98 para poder realizar un comentario se debe ingresar en la sugerencia, poder observar la lista de comentarios y llenar el formulario para generar uno nuevo. Es muy simple, solo se escribe el comentario y se puede calificar del 1 al 5, para indicar respalda la sugerencia. Hay que destacar, que un usuario puede eliminar o editar solo sus propios comentarios.

Figura 4.98: Crear, editar o eliminar comentario

4.8.3. Codificación

La creación de comentarios se realizó mediante el uso de "Scaffolding" de Rails. Con esto obtenemos la estructura básica para listar, mostrar, crear, editar y eliminar comentarios. La única diferencia es que en este caso, todas las operaciones de comentarios se encuentran inmersas en la vista de sugerencias.

Al momento de visualizar una sugerencia debe hacerse una búsqueda de los comentarios que tenga relacionados como se muestra en la Figura 4.99. La edición se realiza en una vista

aparte y la eliminación si se ejecuta desde la misma vista de sugerencia.

```
def show
      add_breadcrumb "Datos de Sugerencia", :sugerencia_path
  @sugerencia = Sugerencia.find(params[:id])

  sql = "SELECT * FROM comentarios WHERE sugerencia_id = #{params[:id]}"
  @comentarios = Comentario.find_by_sql(["#{sql}"])
  @comentarios =  Comentario.paginate_by_sql(sql, :page => params[:page], :per_page => 10)
  @comentario = Comentario.new

  respond_to do |format|
    format.html # show.html.erb
    format.json { render :json => @sugerencia }
  end
end

# GET /sugerencias/new
# GET /sugerencias/new.json
def new
  add_breadcrumb "Nueva Sugerencia", :new_sugerencia_path
  @sugerencia = Sugerencia.new

  respond_to do |format|
    format.html # new.html.erb
    format.json { render :json => @sugerencia }
  end
end
```

Figura 4.99: Código para mostrar sugerencia

Hay que destacar que al listar las sugerencias también hay información que viene dada por los comentarios, como la calificación (estrellas) que resalta el respaldo de una sugerencia.

4.8.4. Pruebas

Se muestran las pruebas de esta iteración.

#	DESCRIPCIÓN DEL CASO DE PRUEBAS	RESULTADO ESPERADO	RESULTADO OBTENIDO
25	Se deben poder generar comentarios de las sugerencias	Se debe poder crear, editar y/o eliminar un comentario de sugerencia.	Los resultados son los esperados.

26	No se debe poder editar o eliminar comentarios de las sugerencias que no sean propias	Se debe poder editar y/o eliminar un comentario de sugerencia si lo hizo el mismo usuario.	Se logra editar y/o eliminar un comentario propio. Sólo se pueden ver los comentarios de otros usuarios, no se pueden editar ni eliminar.

Cuadro 4.17: Pruebas. Iteracion 8.

4.9. Iteracion 9: Profesores invitados y difundir libros sugeridos.

4.9.1. Planificación

Se muestran las historias de usuario de esta iteración.

ID	FECHA	DESCRIPCIÓN
26	22-01-2014	Se debe poder crear un profesor invitado con privilegios limitados, y que solo pueda realizar comentarios sobre las sugerencias
27	22-01-2014	Se puede habilitar la funcionalidad de mandar correos electrónicos una vez creada una sugerencia para poder difundir a ciertos profesores dicha sugerencia.

Cuadro 4.18: Historia de Usuario. Iteracion 9.

4.9.2. Diseño

Para crear un profesor invitado (Cuadro 4.18: ID 26), se despliega un formulario como se observa en la Figura 4.100, que permite capturar datos necesarios y así ingresar un usuario en el sistema. Es necesario conocer nombre, correo y dependencia, luego se enviará la información para autenticarse al correo electrónico proporcionado.

Figura 4.100: CrearProfesorInvitado

Los privilegios de este tipo de profesor son muy limitados, tan solo puede visualizar sugerencias y dentro de ellas realizar comentarios. No puede realizar sugerencias

Para crear el listado de correos electrónicos de profesores que podrían hacer comentarios sobre un libro sugerido (Cuadro 4.18: ID 27), se habilitó un campo en el formulario de sugerencias, en el cual podrán agregarse los correos separados por coma (,) y luego desde la lista de sugerencias se tendrá la opción de enviar el email.

4.9.3. Codificación

Para crear un profesor invitado (Cuadro 4.18: ID 26) el formulario es muy básico por lo que algunos datos deben crearse en el controlador. “Cedula” y “password” serán creados aleatoriamente, las características de la “cedula” tiene un formato parecido a este “guest_XXXX” donde XXXX son dígitos aleatorios, como se muestra en la Figura 4.101.

```
def crear_profesor_invitado

  @usuario_repetido = Usuario.where(:correo => params[:usuario]['email'])

  if !@usuario_repetido.nil?
    redirect_to ppal_profesores_admin_nuevo_profesor_invitado_url,
      :notice => 'Ya hay un usuario con ese correo electronico en el sistema.'  and  return
  end

  @usuario = Usuario.new(params[:usuario])

  @clave_plana_aleatoria = (rand * 9999999).ceil # genera la clave aleatoria
  @usuario.cedula = "guest_#{(rand * 9999).ceil}"
  @usuario.password_digest = Usuario.encriptar(@clave_plana_aleatoria)

  ClaveMailer.clave_usuario_guest(@usuario,@clave_plana_aleatoria,@usuario.cedula).deliver
  respond_to do |format|
    if @usuario.save

      guardar_log(session[:usuario_id], self.class.name,__method__.to_s, @usuario,nil )
      format.html { redirect_to ppal_profesores_admin_nuevo_profesor_invitado_url,
                  :notice => "#{@usuario.nombre} fue creado exitosamente y se le ha
                      enviado su acceso al correo proporcionado." }
    else
      format.html { render :action => "nuevo_profesor_invitado" }
      format.json { render :json => @usuario.errors, :status => :unprocessable_entity }
    end
  end

end
```

Figura 4.101: Código para guardar Profesor Invitado

En la vista principal de profesores, si ingresa un profesor “invitado” se dejan de mostrar del menú todas las opciones excepto las sugerencias. Dentro de las sugerencias la opción de

crear también esta deshabilitada.

4.9.4. Pruebas

Se muestran las pruebas de esta iteración.

#	DESCRIPCIÓN DEL CASO DE PRUEBAS	RESULTADO ESPERADO	RESULTADO OBTENIDO
27	Se crea un profesor invitado desde el módulo de profesores.	Ingresa como profesor (Coordinador o Normal) y se crea un profesor invitado.	Una vez se ingresa como profesor se crea un profesor invitado agregando su correo electrónico y esto genera un contenido con una cedula de invitado y clave para acceder al sistema.
28	Se hacen comentarios y se prueba restricciones de edición y eliminación de un profesor invitado.	Al ingresa como profesor invitado no se puede eliminar ni editar comentarios que no sean propios.	Una vez se ingresa como profesor invitado se puede visualizar las sugerencias hechas. Al ver cada sugerencia no existe la opción de editar y eliminar comentarios de otros usuarios. Tampoco aparece el botón para crear una sugerencia.

29	Se prueba una lista de correos a los cuales se manda un enlace para poder ver una sugerencia.	Se debe poder agregar una lista de correos electrónicos y enviar el contenido por correo.	Una vez creada o al editar una sugerencia se agregó una lista de correos electrónicos, se guardó, esto habilito y ejecuto la funcionalidad de enviar por correo y llego el correo.

Cuadro 4.19: Pruebas. Iteracion 9.

Conclusiones

Este Trabajo Especial de Grado fue el resultado del estudio de los diversos servicios que presta la aplicación Web de la Bolsa del Libro, la solución a ciertas deficiencias del sistema, el desarrollo de nuevas funcionalidades y el diseño e implementación de un nuevo módulo de profesores.

Esta nueva versión del sistema, permite a los administradores visualizar estadísticas actuales e históricas, revisión de reportes donde tendrán la posibilidad de filtrar información para mejorar el enfoque, manejar privilegios de usuarios dando la posibilidad de crear un administrador con menos jerarquía o privilegios y visualizar información sin poder modificarla. Entre otros avances, hay que resaltar la posibilidad que tienen ahora los usuarios del módulo de administración para cambiar o modificar ejemplares antes de ser entregados los libros. Estos procesos no se habian contemplado al momento de automatizar los servicios, pero durante el proceso manual seguramente eran tan obvios que se pasaron por alto.

Se realizaron una serie de cambios que permitían visualizar más información en los reportes, realizar búsquedas por nuevos parámetros, acomodar formularios y restringir la solicitud de solvencias por parte de los estudiantes si todavía poseen libros alquilados.

Para complementar el sistema se desarrolló un módulo de profesores siguiendo un modelo flexible que permitiese realizar mantenimientos correctivos, adaptativos durante la captura de necesidades.

Principalmente fue diseñado para almacenar las sugerencias de los profesores sobre los libros que deberían ser adquiridos por la Bolsa del Libro, luego se agregó una sección que permite ingresar las materias que se dictan en la facultad anexando a cada una la lista de libros recomendados que puedan ser alquilados en la Bolsa.

Se crearon dos tipo de jerarquía de profesores (normal y coordinador), que actualmente no tienen diferencia pero en algún momento lo tendrán.

Las sugerencias que los profesores realicen las podrán visualizar los administradores, sin embargo para dar mayor información sobre dicha propuesta los demás profesores pueden realizar comentarios refutando o apoyando la necesidad de adquirir dicho libro, inclusive se puede estimar del 1 al 5 que tan importante es la sugerencia.

Para conocer la opinión de otros profesores que no se encuentran en el sistema pero se presume pueden ser de gran ayuda al momento de calificar un libro sugerido, se agregó la posibilidad de enviar dicha sugerencia a uno o varios contactos por vía correo electrónico y ellos ingresarán a la aplicación como profesores "invitados" para dar su opinión. Este nuevo tipo de profesor se creó con privilegios mínimos en el módulo de profesores pero le permitirá visualizar las sugerencias y dar su opinión sobre ellas. La creación de este nuevo profesor, se realiza a través de un formulario, el cual lo agrega al sistema como usuario y le suministra toda la información de ingreso por vía correo electrónico.

Para finalizar podemos certificar que los nuevos desarrollos de la aplicación, permitieron actualizar los principales servicios ofrecidos en la Bolsa del Libro, satisfaciendo las necesida-

des actuales de los usuarios a nivel administrativo, estudiantil y docente.

Recomendaciones

En el área de administración para conocer si un estudiante esta solvente se busca que no tenga ningún ejemplar alquilado, sin embargo sería mejor si al pedir la cedula del usuario el sistema respondiera si este estudiante está en deuda o no, y por medio de un link poder ver su historial de alquileres.

Añadir nuevas funcionalidades al Módulo de Pre-Alquiler, permitiendo que en el mismo se efectúe la asignación de los ejemplares solicitados por los estudiantes, tomando en cuenta un estudio socio-económico que facilite dicha asignación.

Implementar el Módulo de Venta de Libros. Existe una gran cantidad de libros en la Bolsa del Libro que se deberian vender para obtener fondos. No solo es necesario los fondos sino que tambien hay necesidad de espacio para nuevos libros.

Buscar la forma de no tener que imprimir un comprobante, sino que se trabaje con información digital y se mantenga la confiabilidad.

Por último, se plantea la idea de crear un nuevo módulo, o área donde se puedan ingresar libros digitales (Español o Ingles).

Bibliografía

[1] Araujo María y Pereira Cidalia, Desarrollo de aplicaciones de comercio electrónico usando software libre. Caso de estudio: Bolsa del Libro de la Facultad de Ciencias de la UCV. T.E.G., Septiembre 2007. 9

[2] Jiménez Andreina y Oviedo Joselyn, Diseño e Implementación de una Aplicación Web para la automatización de los procesos de la Bolsa del Libro. T.E.G., Septiembre 2011. 9

[3] Pernía Arcadio A. y Capote Kenny G., Desarrollo de un Sistema Web para El Control de Los Procesos de La Bolsa Del Libro. T.E.G., Octubre 2012. 4, 9

[4] Ruby S, Thomas D, Hansson D., Agile Web Development with Rails. Dallas, Texas: The Pragmatic Bookshelf., (2011). 39

[5] Ireport Getting Started, Obtenido el 14 de julio de 2013, desde http://community.jaspersoft.com/ 46

[6] Bridge from RoR to Jasper, Obtenido el 18 de Julio de 2013, desde http://rubygems.org/gems/jasper-bridge/.

[7] Danciu Teodor and Chirita Lucian., The Definitive Guide to JasperReports, Apress.,(2007). 45

[8] Acerca de Ruby., Consultado el 02 de Julio de 2014, desde www.ruby-lang.org/es/about/ 40

[9] Comenzando con Rails. Componentes., Consultado el 10 de Octubre de 2014, desde http://gomix.fedora-ve.org/projects/ruby/wiki/Comenzando_con_Rails-componentes/ 41

[10] IBM., Guía del Estudiante Base de Datos I., 2007. 43

[11] Extreme Programming. Gentle Introduction., Consultado el 02 de Julio de 2013, desde http://extremeprogramming.org/

[12] Pressman Roger S., Ingeniería del Software. Un enfoque práctico. McGraw Hill Interamericana, sixth edition, 2006. 51, 52, 53

Apéndice

Para el estudiante que continue con el desarrollo de los procesos en la Bolsa del Libro aquí están algunos consejos:

Hay que tener en cuenta que el uso de Git como Sistema Controlador de Versiones o VCS "version-control system", viene acompañado de una GEMA llamada "Capistrano" que permite subir de manera más simple los cambios al servidor.

La secuencia que permite esto es básicamente la siguiente:

- **git add .** : detecta los cambios hechos
- **git commit**[Aquí colocar información de los cambios que se van a subir] : Se prepara para subir los cambios detectados y le asigna una información como etiqueta.
- **bundle pack** : Se detectan las gemas usadas en el area de desarrollo.
- **git push** : Se encarga de subir los cambios al servidor.
- **cap deploy** : Este último comando provcara que es sistema deje de estar activo, para solucionar esto, por medio de la consola debemos ubicarnos en el directorio que con-

tiene los cambios que se acaban de subir al servidor y darle los permisos 777.

La dirección en el servidor es: **cd ../home/rubys/work/bolsadellibro/**

Tanto para hacer push como para entrar en el servidor se requieren claves de acceso. Estas seran entregadas en su momento por el Profesor a cargo.

JasperReport: esta herramienta permite hacer los reportes de la Bolsa del Libro. Se recomienda instalar ireport y junto a este una version de Java.